VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro identifikaci chování studenta v průběhu on-line zkoušení

System for student's activities identification during test progress

2009 **Ondřej Bialas**

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal. Beru na vědomí, že Vysoká škola báňská – technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě bakalářskou práci užít (§35 ods. 3).

V Ostravě dne ……. …….
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Radoslavu Fasugovi za účinnou pedagogickou a odbornou pomoc a všechny další cenné rady při zpracování této bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá především návrhem a následnou implementací systémů pro sledování studenta v průběhu online zkoušení a následného vyhodnocení jeho chování. Úvod tohoto textu pojednává o problematice podvádění studentů při online testech a o možných opatřeních pro prevenci a odhalení nežádoucí činnosti. Od třetí kapitoly je pozornost věnována použitým či vyzkoušeným metodám pro sledování aktivit studenta a také možným způsobům odeslání zachycených dat na centrální úložiště. Další část práce je věnována principu vyhledávání ikon v obraze mnou navrhnutým algoritmem. Poslední kapitoly této práce se zabývají návrhem struktury aplikací a jejich experimentálním testováním po implementaci.

Klíčová slova

Alfa kanál, formáty obrázků, hledání změn v obraze, online testy, pixel, procházení obrázků, předzpracování obrázku a ikon, XML protokol.

Abstract

This bachelor thesis particulary deals with design and subsequent implementation of a system for monitoring the student during the online examination and subsequent evaluation of his behavior. Introduction of the text deals with the issue of students cheating in online tests and with possible provisions to prevent and to detect of undesirable activities. From the third chapter, attention is drawn to used or proven methods for the monitoring of student activities and possible ways to send the captured data to a central repository. Another part of the work is devoted to the principle of searching icons in the screen with algorithm designed by me. The last chapter of this work deals with the design of the structure of applications and their experimental testing after the implementation.

Keywords

Alpha channel, image formats, searching image changes, online tests, pixel, image browsing, image and icon preprocessing, the XML protocol.

Seznam použitých symbolů a zkratek

API - Application programming interface
BMP - Microsoft Windows Bitmap
CPU - Central processing unit
FTP - File transfer protocol
GIF - Graphic Interchange Format
GrayScale – Šedá škála
GUI - Graphical User Interface
HTML – HyperText Markup Language
HTTP - Hypertext Transfer protokol
ICQ - I seek you
IP - Internet Protocol
JPEG - Joint Photographic Experts Group
JS – JavaScript
LZW - Lempel Ziv Plch
MSN - Microsoft Network
PNG - Portable Network Graphics
PX – Pixel
RD - Remote Destktop Connection
RGB – Red, Green, Glue
SOAP - Simple Object Access Protocol
TIFF - Tag Image File Format
UDDI - Universal Description, Discovery and Integration
UML - Unified Modeling langure
VNC - Virtual Network Computing
W3C – World Wide Web Consortium
WSDL – Web Service Description Language
WSIL - Web Services Inspection Language
XML - Extensible markup langure
XMPP - Extensible Messaging and Presence Protocol
XSL – Extensible stylesheet language

OBSAH

1. Úvod

S rozvojem výpočetní techniky roste i počet možných oblastí jejího využití. Počítače se používají prakticky k čemukoliv a kdekoliv. Jinak tomu není ani ve školství a asi nikdo si už nedokáže představit studium bez počítačů a internetu.

Stále častěji se také objevují zkoušky, písemky či testy, které se provádí právě na počítači. Pokud si však představíme učebnu plnou počítačů (Obrázek 1), kde za každým sedí student a snaží se vykonat danou online zkoušku je jasné, že se snad každý při dnešních nejen technických možnostech bude pokoušet ovlivnit výsledek online zkoušky v jejím průběhu ke svému prospěchu.

Výsledkem této práce by mělo být odhalení možných podvodů neboli zaznamenávání a následné vyšetření chování studenta v průběhu online zkoušení. Tohoto by mělo být docíleno pomocí 2 programů. První z nich se bude nacházet na straně počítačů, na kterých bude online zkoušení probíhat. V jeho průběhu bude ukládat obrázky toho, co má student právě na své obrazovce. Druhý program pak umožní zkoušejícímu z těchto obrázku zjistit, zda daný student prováděl během zkoušení na svém počítači nějaké zakázané aktivity.

Obrázek 1: Učebna

2. Nástin problematiky

Kvůli rostoucímu počtu studentů jsou ústní zkoušky stále více na ústupu a přechází se buď k písemným či online testům. Dohled při nich je minimální.

2.1. Online testy

Samotné online testy nedokážou zcela zabránit tomu, aby při jejich průběhu studenti nějakým způsobem podváděli. Avšak vývojáři či autoři těchto testů, se snaží možnost podvádění alespoň snížit například těmito způsoby:

a) Načasováním testu - vyučující pak smí určit čas, od kdy bude test pro studenty zpřístupněný. Tj. před tímto časem nebude moci žádný student test spustit. Dále smí určit čas, do kdy test musí být ukončen, jinak se buď ukončí automaticky, nebo smí student pokračovat, ale otázky zodpovězené po zadaném čase ukončení již nebudou bodovány. Nastavením těchto časů tedy lze nastavit časový limit, kterým lze studentům v podstatě minimalizovat podvádění. V krátkém časovém limitu studenti nebudou mít možnost podvádět.

b) Seznamem úloh – vyučující smí stále přidávat nové otázky úloh a také možné odpovědi k nim, které pak student může mít v testu zadány. S velkým počtem otázek a různých možných správných a špatných odpovědí na ně roste počet kombinací, jak otázky s odpověďmi zamíchat. Tj. každý test může mít jiné otázky úloh a k nim jiné možnosti odpovědí. Tímto nastavením můžeme trochu ztížit studentům možnost opisování. Studenti však mohou mít na absolvování testů obvykle více pokusů, pro které je vhodné, aby se každý pokus zakládal na předešlém. Témata otázek testu by měla být podobná tématům z testu předešlého.

c) Nezobrazením výsledků – po ukončení testu se studentům nezobrazí jejich špatné ani správné odpovědi. Toto snižuje možnost vytvoření „taháku“ se správnými či špatnými odpověďmi pro další testy v daném předmětu s možným vygenerováním stejných zadání některých otázek a příslušných odpovědí.

d) Ověřením údaje – zde patří především dvě možnosti. První z nich omezením přístupu k testu jen pro určité počítače neboli počítačům z určité podsítě či jen konkrétnímu počítači s konkrétní IP adresou. Na počítači s „nepovolenou“ IP adresou nesmí být test spuštěn. Druhou možností je, že test smí být spuštěn jen tehdy, pokud je správně zadáno heslo pro daný test. Tímto lze dosáhnout především toho, že k testu bude mít přístup jen vybraná skupina studentů, které bude heslo sděleno například těsně před testem v dané učebně.

e) Ostatním zabezpečením – je možné, aby se test spustil v okně, ve kterém budou omezeny některé funkce, které studentům běžně nabízí internetové prohlížeče. Jako příklad si můžeme uvést kopírování či vkládání textu z jiného souboru.

2.2. Co systém nebude či nemůže řešit

Způsobů, jak podvádět a opisovat při testech, je samozřejmě mnoho. Určitě tedy nelze podvádění zcela zabránit a bohužel některým způsobům podvádění zabránit ani nemůžeme. Dnes se podvádí velmi rafinovaně, a ač to může znít troufale, tak při online testech i ve velkém.

Prvním a hned velmi extrémním podvodem, kterému nemůžeme zabránit je zneužití identity, tzn., že za daného studenta vykoná test někdo jiný. Padělat průkaz, kterým se vždy před testem musí každý student prokázat, není dnes jistě nepřekonatelný problém. Bohužel i takové případy se stávají. Ještě větším extrémem je, pokud student z nějakých důvodů má povolení a provádí test dálkově bez jakéhokoliv dohledu. Zde opravdu nemůžeme nikdy dosáhnout stoprocentní jistoty, že test vykonal opravdu ten student, za koho se vydává.

Další možnost podvodu spočívá v použití tzv. taháků, které nejsou v elektronické podobě na počítači. Papírové taháky jsou již dnes méně využívány a studenti jdou v podvádění s dobou. Testy nebo správné a špatné odpovědi možných otázek k nim od různých zdrojů si fotí mobilními telefony a tyto fotky si pak rozšiřují mezi sebou. Při případném opakování testu pak fotky používají jako tahák a pokud nejsou přistiženi vyučujícím dohlížejícím na průběh testu, nemůžeme tento podvod zachytit. Velké množství otázek a generování různých odpovědí k nim jen omezuje tuto možnost podvodu.

2.3. Specifikace zadání

První část práce bude tvořit aplikace, která bude sloužit jako prostředek pro sledování chování studenta během online testů. Toho bude docíleno na základě zaznamenávání stavu pracovní plochy a hledání změn v ní, alternativně pak pomocí sledování procesů. Po každém testu tedy musí mít vyučující (zkoušející) k dispozici od každého studenta sadu obrázků zachycených během testu a k nim příslušné informace v protokolu. Je tedy nutné zajistit přenos zachycených dat od každého studenta směrem k vyučujícímu nebo do jistého centrálního úložiště. Aplikace nesmí nijak omezovat studenta při vykonávání testu a neměla by zachytávat velké množství irelevantních dat.

Druhou část bude tvořit aplikace pro vyhodnocení chování studentů pomocí obrázků zachycených při jejich testech a zobrazení důležitých informací z protokolu. Stěžejním bodem této části bude vyhledání předem připravených ikon, které by se neměly na žádném z obrázků vyskytovat s jejich následným zvýrazněním pro vyučujícího. Důležité je, aby vyhodnocení obrázků studentů proběhlo co nejrychleji, je tedy nutné nalézt co nejrychlejší algoritmus pro vyhledávání ikon (rozpoznatelných fragmentů nepovolené činnosti). Systém bude umět provádět jak kompletní vyhledávání, kdy v každém obrázku nalezne všechny výskyty zakázaných ikon, tak rychlé vyhledávání, kdy po prvním nalezení libovolné ze zakázaných ikon okamžitě přejde na další obrázek. Dále je požadována možnost vyhodnocení několika studentů najednou nebo vyhodnocení chování pouze jediného studenta.

2.4. Plánované cíle

Hlavním cílem této práce je, jak už bylo řečeno v úvodu, odhalení zakázaných aktivit v průběhu online testů. Zakázané aktivity budou pro nás nepovolené činnosti na počítači, které lze relativně snadno zjistit podle toho, co vše měl student během testu na obrazovce či jaké měl spuštěné procesy. Nyní si uvedeme pár příkladů zakázaných aktivit, které bychom měli být schopni odhalit:

a) Procházení internetu, použití elektronické pošty. Student si hledal na internetu možné odpovědi k otázkám testu.

b) Velmi oblíbené je dnes při online testech používat e-mailu a komunikačních klientů podporujících různé protokoly jako ICQ, MSN, XMPP atd., které nabízí možnost konzultovat jednotlivé otázky.

c) Prohlížení pomocných studijních materiálů na flash discích a jiných externích paměťových mediích.

d) Test byl prováděn pomocí funkce vzdálené plochy, která umožňuje vzdálené řízení počítače. A to buď standardní RD funkce (Remote Destktop Connection) ve Windows, kde je plocha přenášena jako objekty nebo VNC (Virtual Network Computing) založené na přenášení vzdálené plochy jako bitmapy.

Snahou bude všechny tyto možné podvody odhalit a penalizací strhnout studentovi část z dosažených bodů, případně ho pak nechat test zopakovat. Z hlediska legislativy by student, který odmítne sledování jeho chování během testu pomocí námi vytvořené aplikace, musel požádat o speciální dozor. Všichni ostatní studenti, kteří budou souhlasit s jejich sledováním pomocí aplikace, si snad jen při pomyšlení, že probíhá sledování jejich aktivit na počítači, rozmyslí, zda se pokusí podvádět.

2.5. Volba implementačního prostředí

Systém bude vyvíjen pro operační systém MS Windows XP a Vista. Aplikace budou desktopové. Proto budou zdrojové kódy aplikací psány v jazyce C-sharp (C#), který je jednoduchý, moderní, mnohoúčelový a objektově orientovaný programovací jazyk vyvinutý firmou Microsoft jako jazyk platformy .NET. Vhodný pro desktop aplikace ve Windows. Jazyk C# je také integrován do vývojového prostředí Visual Studio 2008 .NET [1], kterého bude pro implementaci využito.

3. Hlídání studenta

Prvotní idea, jak hlídat studenta při testu, spočívala pouze v kontrole stavu aktivního okna, ve kterém jsou vyplňovány otázky testu. Pokud by se otevřené okno testu v jeho průběhu stalo neaktivní, znamenalo by to, že se student pokouší o nedovolené činnosti a následné vykonávání testu by mohlo být například zablokováno. Toto řešení by bylo ovšem velmi radikální především pro případy, že se okno stane neaktivní ne studentovou vinou (např. při aktualizaci antiviru se otevře nové okno a okno testu se stane neaktivní). Je tedy vhodné sledovat aktivity studenta za pomocí snímání pracovní plochy obrazovky.

Jistě by šlo v průběhu online zkoušení snímat obrazovku v co nejmenších intervalech, každý obrázek uložit a pak z kolekce těchto obrázků vyhodnotit chování daného studenta. To však nemá smysl, především ne tehdy, pokud zrovna na obrazovce nedochází po delší dobu k žádným změnám, nedojde ke spuštění nového procesu, nezmění se činnost procesoru nebo je využití paměti stále stejné. Dostali bychom tak spoustu irelevantních dat. Dále je tu samozřejmě nevýhoda toho, že velké množství uložených obrázků představuje velký objem dat, který je nutno odeslat do vzdáleného úložiště a nakonec vyhodnotit.

Základní dva přístupy pro hlídání studenta jsou tyto:

a) Hlídání podle spuštěných procesů, využití procesoru a paměti

b) Hlídání podle obrazovky, hledání změn v obraze

3.1. Hlídání podle spuštěných procesů, využití procesoru a paměti

U tohoto přístupu se po spuštění aplikace nejprve sejme obrazovka a obrázek ihned uloží. Poté bude aplikace neustále kontrolovat spuštěné procesy vyjma naší aplikace. Tato kontrola se zaměří na tyto případy:

a) Byl spuštěný nový proces

b) Byl ukončený nějaký proces

c) Některý z procesů začal méně či více vytěžovat procesor

d) U některého z procesů se změnila hodnota využití paměti

Pokud náš systém zjistí, že nastal jeden z těchto případů, nasnímá sekvenci obrázků a všechny uloží. Každý obrázek této sekvence bude sejmut po jiném časovém úseku. Jak by sejmutí takovéto sekvence obrázků mohlo vypadat, ukazuje následující tabulka (Tabulka 1):

Obrázek	Čas uplynulý od sejmutí prvního obrázku
č.2	1 sekunda
č.3	1 sekunda
č.4	2 sekundy
č.5	4 sekundy
č.6	6 sekund
č.7	9 sekund
…	…

Tabulka 1: Časové odstupy mezi sejmutím obrazovky

Je však jisté, že není důležité ani vhodné hlídat vytížení CPU i využití paměti u každého procesu. Je zbytečné, aby například byla nasnímána sekvence obrázků po pohybu kurzoru myši. S velkou pravděpodobností není potřeba hlídat všechny systémové procesy. Důraz bude kladen na hlídání procesů, jejichž vlastníkem je právě přihlášený uživatel. Zcela nejvyšší prioritu budou mít procesy internetových prohlížečů.

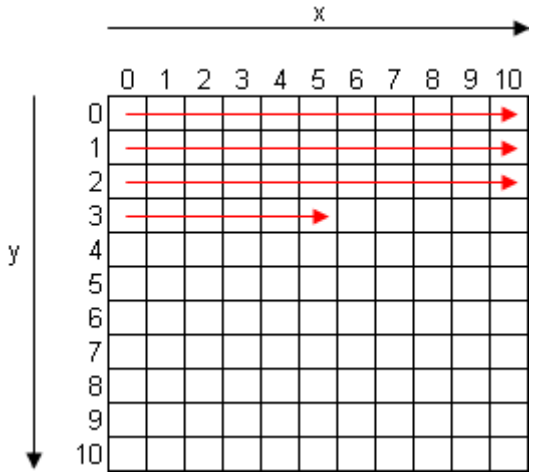
Velká nevýhoda tohoto přístupu se projeví při online testech používající pro otázky či úkoly client-side dynamický web. Jako příklad si můžeme uvést úlohu, kde zadáním je nakreslení určitého diagramu či schématu a student musí na plátno zobrazené ve svém prohlížeči tento diagram nakreslit. Pro všechny client-side technologie jako je Macromedia Flash, JavaSript, Java Applet, ActiveX atd. pro tvorbu dynamického webu platí, že aplikace jsou spuštěny až na straně klienta po jejich stažení (kód je součástí HTML dokumentu a provádí se na straně uživatele). To znamená, že procesor počítače, na kterém je test prováděn, bude u těchto úkolů neustále pracovat a měnit své vytížení a bude tak nasnímáno a uloženo množství nadbytečných obrázků.

3.2. Hlídání podle obrazovky, hledání změn v obraze

Stejně jako v předchozím případě systém po spuštění hlídání sejme obrazovku a tento první obrázek okamžitě uloží. Po vhodně zvoleném intervale dojde k dalšímu sejmutí obrazovky a systém podle níže uvedených metod zjistí, zda daný obrázek uloží nebo je zcela irelevantní jej ukládat. Pokud dojde k uložení, bude tento obrázek sloužit pro další porovnávání místo původního. Interval mezi sejmutím obrazovky bude zvolen tak, aby student neměl možnost v tomto čase například rychle nahlédnout na nějaké internetové stránky, přečíst potřebné informace a stránky zavřít. Standardně bude tento interval nastaven na 3 až 4 sekundy a bude jej možné konfigurovat.

3.2.1. Procházení obrázků po pixelech

Každý rastrový obrázek se skládá z jednotlivých obrazových bodů – pixelů uspořádaných v matici. Nejjednodušší metodou, jak zjistit zdali se dva obrázky mezi sebou liší, je procházet matice po pixelech (Obrázek 2) a tyto pixely porovnávat. Pokud budeme porovnávat dva obrázky s vysokým rozlišením, je jasné, že půjde o časově náročnou operaci. Složitost této operace je n (n = počet pixelů obrázku). Také pro nás nemá smysl u obrázků porovnávat, zdali se liší v každém pixelu. Každá podstatná změna v obraze, kterou systém musí zachytit, sebou nese změnu v několika pixelech vedle sebe. Především z těchto důvodů, většinou při nějaké změně obrazovky, stačí procházet a porovnávat pixely obrázků s určitým skokem (porovnávat například každý třetí pixel).

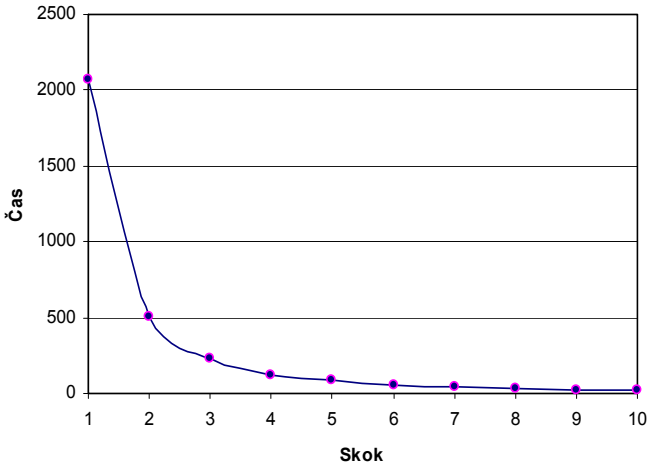


Obrázek 2: Směr procházení pixelů obrázků

V následující tabulce a grafu (Tabulka 2, Obrázek 3) můžeme vidět průměrnou dobu porovnání dvou náhodných obrázků po různých skocích v rozlišení 1280x800 na počítači s procesorem Intel Core2 Duo 2.0GHz.

Skok procházení	Čas [ms]
1	2067
2	510
3	232
4	125
5	83
6	56
7	44
8	33
9	26
10	20

Tabulka 2: Testování skoků



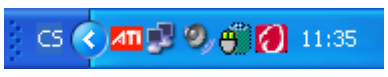
Obrázek 3: Graf porovnávání rychlostí procházení

Z této tabulky a grafu vyplývá, že z důvodu rychlosti běhu programu je při porovnávání celých obrázků pixel po pixelu zcela nepoužitelné a je vhodné zvolit skok větší než 3. Tj. v nejhorším případě se bude porovnávat každý čtvrtý pixel.

3.2.2. Změny v hlavním panelu

Změny v hlavním panelu jsou při hledání změn na obrazovce v systému stěžejním bodem. Sebemenší změna v hlavním panelu může pro náš systém znamenat, že student v době testu provádí na svém počítači nějakou podezřelou činnost a je tedy nutné obrazovku v době této změny sejmout a obrázek následně uložit. Porovnávání obrázků v oblasti hlavního panelu bude systém provádět pixel po pixelu.

Tray-ikony nacházející se v hlavním panelu v tzv. oznamovací oblasti, mají v průměru rozměry 15x16 pixelů. To znamená, že pokud nalezneme v hlavním panelu rozdíly ve 240 pixelech, mohl student něco spustit a náš systém by to měl zaznamenat. Většinou se však spuštěním nějaké aplikace, která má svou tray ikonu posune i velikost oznamovací oblasti hlavního panelu a to sebou přináší změnu ve více pixelech. Systém tedy bude zaznamenávat změny v hlavním panelu nad 400 pixelů, avšak je vhodné, aby bylo možné tuto hodnotu konfigurovat. Ukázku oznamovacích oblastí hlavního panelu se zapnutým panelem jazyků máme na obrázcích (Obrázek 4 a Obrázek 5):



Obrázek 4: Původní oznamovací oblast, bez zakázaných ikon.

Obrázek 5: Oznamovací oblast se kde došlo ke změně a je nutné uložení obrázku

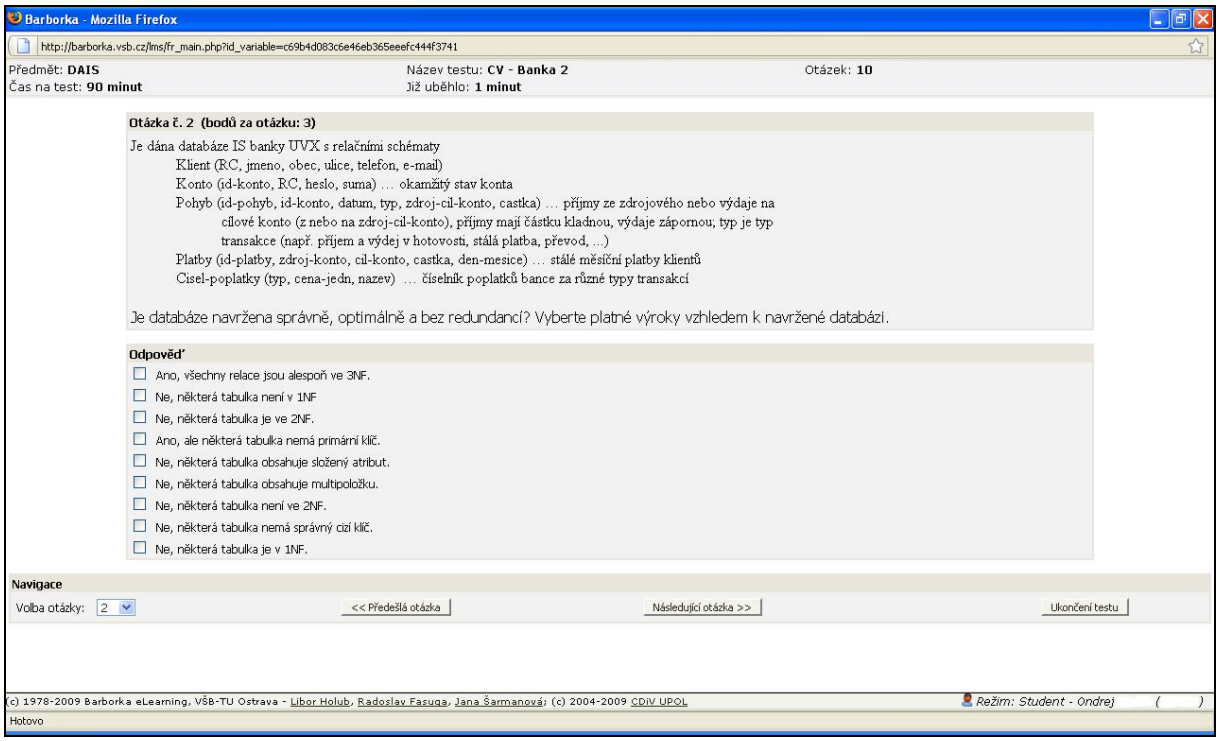
Pozici hlavního panelu či jeho velikost lze v systému Windows samozřejmě měnit. Pokud systém ze systémových informací zjistí, že nastal jeden z těchto případů, sejme obrazovku a obrázek okamžitě uloží.

Poslední případ, který může nastat, je situace, kdy celý hlavní panel může být v průběhu testu skrytý nebo v něm nedochází k žádným změnám. Nemůžeme tedy podle něj zjistit, zdali se má sejmutý obrázek uložit. Tento problém se pokusí řešit následující metody.

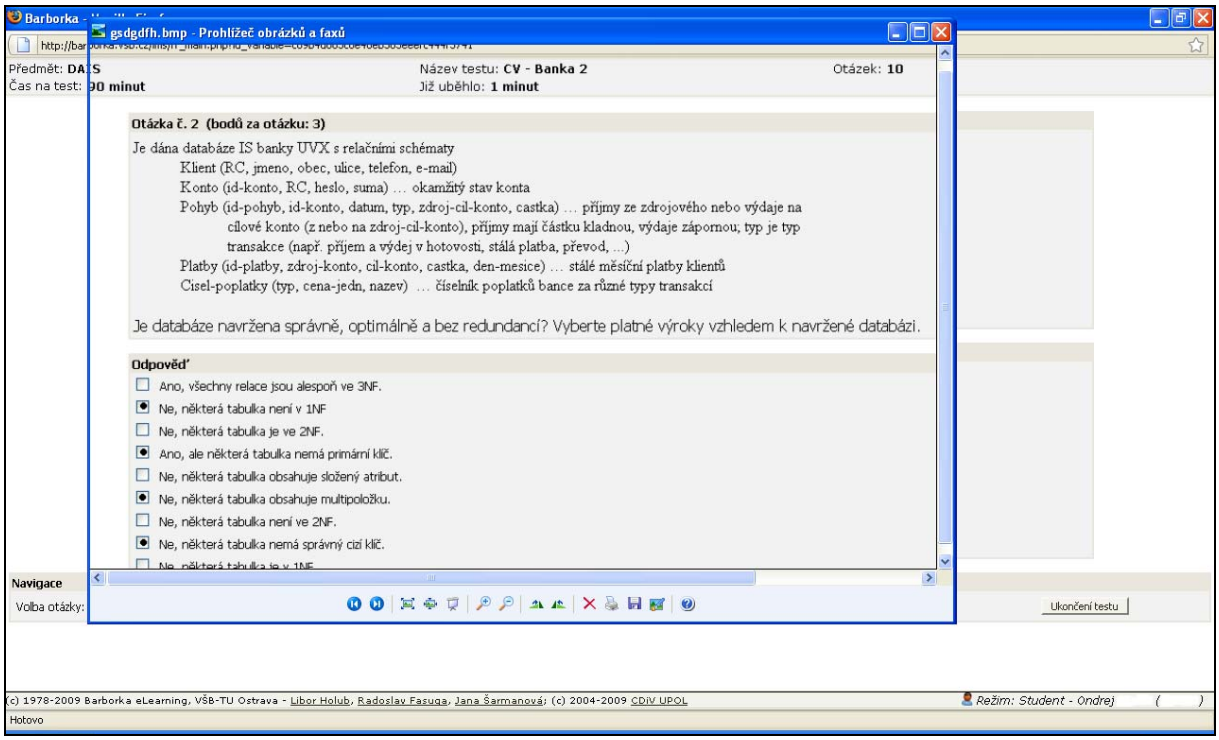
3.2.3. Hledání změn obsahu a okraje okna

Pokud v hlavním panelu nenastanou žádné změny nebo je skrytý, bude systém procházet a porovnávat celé obrázky s vhodně nastaveným skokem. Z kapitoly 3.2.1 víme, že tento skok z důvodu rychlosti běhu programu by měl být větší než 3. Zároveň však nemůžeme nastavit skok příliš velký, aby systém nemohl přeskočit již docela podstatné změny. Skok při procházení nastavíme kompromisně na hodnotu 4 a bude se tak porovnávat každý čtvrtý pixel.

Systém bude mít na vstupu procentuálně (nebo počtem rozdílných pixelů) zadáno, v kolika pixelech se mohou dva právě se porovnávající obrázky lišit. Pokud při procházení s nastaveným skokem a porovnávání příslušných pixelů dvou sejmutých obrázků dojde k překročení této meze, obrázek automaticky uložíme. Pokud však počet neshodných pixelů nepřekročí zadanou mez, nemůžeme obrázek hned zahodit. Může totiž nastat situace, kdy na jednom z obrázků je zachyceno nově otevřené okno, které má podobné složení barev jako to, co je aktuálně na obrazovce pod ním „schováno“ (Obrázek 6 a Obrázek 7). Systém by pak při nastaveném kroku mohl nalézt jen málo změn v obraze a neuložil by obrázek, na kterém mohla být zakázaná aktivita.

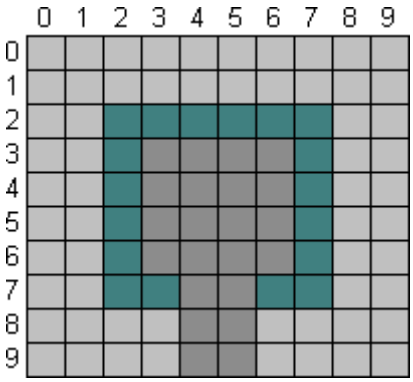


Obrázek 6: Sejmutá obrazovka, bez zakázaných aktivit

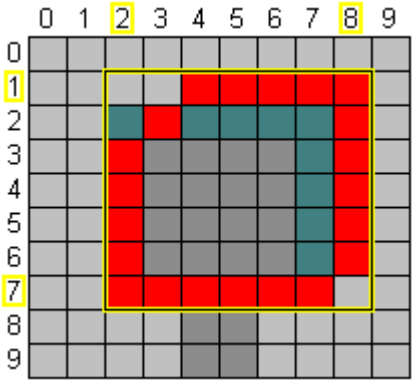


Obrázek 7: Následně sejmutá obrazovka s nově otevřeným oknem

K tomuto účelu bude systém uchovávat nejnižší (nejvyšší) x-vou a k ní příslušnou nejnižší (nejvyšší) y-vou hodnotu souřadnic pixelů, ve kterých se porovnávané obrázky neshodují. Na následujících obrázcích (Obrázek 8 a Obrázek 9) máme ukázku nalezení těchto souřadnic na bitmapách o rozměrech 10x10 pixelů. Nejnižší souřadnice změn budou [2, 1] a nejvyšší souřadnice změn [8, 7], které jsou na obrázku žlutě vyznačeny.



Obrázek 8: Původní obrázek



Obrázek 9: Obrázek se změnami

MS Windows má standardně nastavenou šířku okraje okna 4 pixely. Budeme-li porovnávat dva obrázky, kde na jednom z nich je otevřené okno navíc, nemůže s námi nastaveným skokem procházení nastat situace, kdy přeskočíme pixely okraje tohoto okna. Následně využijeme toho, že na nejnižších souřadnicích neshodných pixelů porovnávaných obrázků by se pravděpodobně mohl nacházet bod levého horního rohu okraje nově otevřeného okna. Poté zkusíme porovnat několik desítek pixelů pod ním tj. pixely se stejnou x-vou a

vyšší y-vou hodnotou souřadnice. Podobně porovnáme pixely vpravo od tohoto bodu. Tj. pixely se stejnou y-vou ale vyšší x-vou hodnotou souřadnice. Pokud se i v těchto pixelech porovnávané obrázky neshodují, je nutné obrázek uložit z důvodů podezření nově otevřeného okna.

3.2.4. Dohledání změn v podezřelé oblasti

Pokud nebude ani po výše zmíněné metodě hledání možného okraje okna obrázek uložen a označen za aktuální pro další porovnání, je možné zjistit počet všech změn v podezřelé oblasti. Tj. porovnáme všechny pixely od nejnižší souřadnice změn do nejvyšší souřadnice změn. Pokud počet neshodných pixelů přesáhne zadanou mez, obrázek uložíme.

3.3. Uložení obrázků

Formátů, pod kterými lze významné (podezřelé) obrázky ukládat, je mnoho. Je potřeba, aby obrázky byly co nejvíce komprimované a měly tak co nejmenší datový objem. Zároveň je nutné, aby komprese byla bezeztrátová a byla tak zachována kvalita obrazu pro pozdější vyhodnocení obrázků. Jako nejvhodnější se jeví formát PNG určený pro bezeztrátovou kompresi rastrové grafiky využívající algoritmus LZ77 [4]. Porovnání komprese PNG s ostatními známými formáty vidíme na následujícím obrázku (Obrázek 10):

Obrázek 10: Porovnání velikosti obrázku uložených do různých formátů

3.4. XML protokol

Zachycené obrázky, které budou ukládány, je potřeba nějakým způsobem zdokumentovat. Jinými slovy řečeno, je nutné pro každý právě uložený obrázek vytvořit v souboru (protokolu) záznam a k němu příslušné údaje. K těmto údajů patří především název obrázku, velikost obrázku, pozice a šířka hlavního panelu, datum a čas uložení obrázku. V době sejmutí prvního obrázku také vždy zjistíme všechny aktuální procesy, URL dokumentů právě prohlížených v internetovém prohlížeči a cesty v souborovém systému právě otevřených oken. Ke každému procesu pak jeho název, vlastníka, využití CPU a paměti. Z důvodů velikosti celého protokolu pak ke všem ostatním obrázkům, které budou systémem ukládány, budeme ukládat jen ty procesy, adresy a cesty, které dosud nebyly studentem spuštěny či otevřeny.

Všechny tyto údaje budou zapsány v jazyce XML (Extensible Markup Languagge). XML byl sestrojen pro popis dat, výměnu dat mezi aplikacemi a publikování dokumentů. Jednou možností je pomocí různých stylů provést transformaci XML do jiného typu dokumentu, nebo do jiné struktury XML. Část XML protokolu vypadá takto:

```
<?xml version="1.0" encoding="utf-16"?>
<protocol>
  <image name="CCCNNN_09,03,19_20,20,18_c0.png">
    <size>138338</size>
    <taskbar width="34">bottom</taskbar>
    <date>09,03,19_20,20,18</date>
    <processes>
      <process process_name="explorer">
        <memory_usage>13908kB</memory_usage>
        <cpu_usage>0%</cpu_usage>
        <user>USER-PC\User</user>
      </process>
      ...
      ...
      <process process_name="Idle">
        <memory_usage>28kB</memory_usage>
        <cpu_usage>93,3333%</cpu_usage>
      </process>
    </processes>
    <paths>
      <path num="1">C:/Documents and Settings</path>
    </paths>
  </image>
  <image name="CCCNNN__09,03,19_20,20,22_c1.png">
    <size>140926</size>
    <taskbar width="34">bottom</taskbar>
    <date>09,03,19_20,20,22</date>
    <processes>
      <process process_name="firefox">
        <memory_usage>43144kB</memory_usage>
        <cpu_usage>0%</cpu_usage>
        <user>USER-PC\User </user>
      </process>
    </processes>
    <addresses>
      <address num="1">http://www.vsb.cz/</address>
    </addresses>
  </image>
  …
  …
</protocol>
```

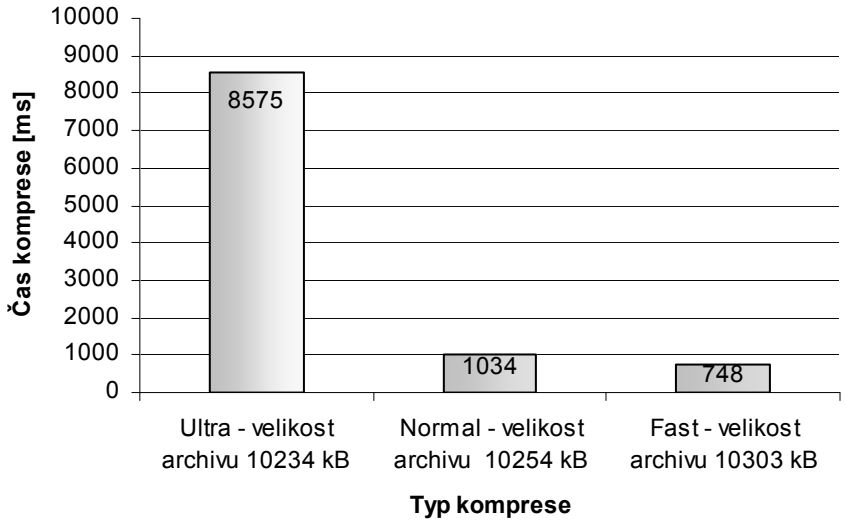
3.5. Archivace

Obrázky v PNG formátu jsou už samy o sobě velmi komprimované. Avšak pokud mají být všechny obrázky spolu s protokolem studenta po skončení testu někam poslány či zkopírovány najednou, je vhodné udělat archiv celé složky studenta. K tomuto účelu bude náš systém využívat freeware komprimační nástroj 7zip. Ten má v porovnání se shareware nástrojem WinRar v průměru až o 10% větší kompresi. V následující tabulce (Tabulka 3) vidíme úroveň „ultra“ komprese složky několika náhodných sejmutých obrázků uložených ve formátu PNG s k nim příslušným XML protokolem:

Počet sejmutých obrázků	Velikost složky s protokolem	Velikost složky po komprimaci	Průměrná komprese na 1 obrázek
3	409 kB	349 kB	20 kB
6	753 kB	681 kB	12 kB
9	1486 kB	1320 kB	18 kB
12	2871 kB	2693 kB	15 kB

Tabulka 3: Úroveň „ultra “komprese

Ultra komprese 7zip má oproti ostatním kompresím velmi výborný kompresní poměr. Toho je však docíleno na úkor doby zpracování. Z tohoto důvodu je vhodné snížit úroveň komprese a kompromisně tak zvolit například rychlou („Fast compression mode“) nebo normální („Normal compression mode“) kompresi, jejichž porovnání s ultra kompresí vidíme na následujícím grafu (Obrázek 11). Komprimována byla složka obsahující 130 obrázků o celkové velikosti 12MB (12293 kB). Z grafu vyplývá, že použití ultra komprese je důvodů rychlosti komprimace zcela nevhodné a co se týče konečné velikosti výsledného archivu i zbytečné.



Obrázek 11: Graf porovnání rychlostí různých úrovní komprese

4. Přenos zachycených dat

Data zachycená u každého studenta je nutné poslat na nějaký server, odkud mohou být stažena či přímo použita pro vyhodnocení. Je důležité, aby studentům bylo co nejvíce znemožněno do zasílaných dat jakýmkoliv způsobem zasahovat.

4.1. FTP odesílání

FTP (File transfer protocol) je protokol aplikační vrstvy z rodiny TCP/IP [2], který umožňuje kompletní práci se soubory na vzdáleném počítači jako je kopírování, mazání, přejmenovávání a v neposlední řadě také vytváření adresářů. V protokolu je použit model klient-server. FTP server poskytuje data pro další počítače. Klient se pak k serveru smí připojit a provádět výše uvedené operace.

Aplikace hlídající studenta může být jednoduchým klientem a posílat zachycená data na předem zadaný FTP server. Nabízí se dvě možnosti odesílání, kterými se zabývají následující 2 kapitoly:

4.1.1. Průběžné odesílání

Při průběžném odesílání by po spuštění aplikace hlídání byla na FTP serveru nejprve vytvořena složka s informacemi o studentovi právě provádějícím test (jméno, IP počítače, název počítače). Následně by všechny sejmuté a podezřelé obrázky byly odesílány na server do příslušné složky v průběhu hlídání. Při ukončení hlídání (po ukončení testu) by pak byl na server odeslán i XML protokol obsahující údaje ke každému obrázku. Nevýhodou při tomto odesílání je, že zkoušku smí v zadaném termínu vykonávat velký počet studentů. Hrozí tak nebezpečí odesílání mnoha souborů v jeden časový okamžik a přetížení serveru.

4.1.2. Odesílání celého archivu

U tohoto přístupu by celá složka studenta se všemi podezřelými obrázky společně s XML protokolem byla při ukončení hlídání zkomprimována do archivu a poté odeslána na server. Výhodnou je, že studenti obvykle neukončí test ve stejnou dobu a odeslání by tak probíhalo v jiných časových okamžicích. Oproti tomu je velkou nevýhodou celková doba archivace složky studenta a následného odeslání velkého objemu dat na FTP server najednou. Archiv studenta, který by se v průběhu testu trvajícího 90 minut pokoušel stále o nějaké zakázané aktivity, může v konečném důsledku dosahovat velikosti řádově i několika GB.

4.1.3. Stažení archivů

Stažení log souborů (archivů s podezřelými obrázky a XML protokolem) jednotlivých studentů pro vyhodnocení z FTP serveru bude provedeno již pomocí nějakého freeware ftp klienta. Jako příklad těchto klientů uvedeme FileZilla, Clear FTP nebo UltraFXP apod.

4.2. Webové služby

Na úvod objasníme pojem webových služeb, jednoduché technologii založené na standardech světové organizace W3C, pyšnící se v současnosti obrovskou popularitou a rozvojem. Webové služby neboli Webservices představují technologii pro vzdálené volání procedur pomocí zpráv v XML jazyce prostřednictvím HTTP protokolu. Nejdůležitější vlastností je úplná nezávislost na platformě tj. na operačním systému, programovacím jazyku atd. Z technického hlediska webové služby využívají architekturu klient-server, kde si klient

společně se serverem vyměňují *SOAP* zprávy (založené na jazyce XML). Zprávy využívají popis vytvořený pomocí *WSDL* (*Web Service Description Language*) jazyka [3]. Technologie se tedy skládá z těchto tří základních částí:

a) SOAP - protokol pro vzdálené volání procedur, pomocí kterého jsou posílány zprávy založené na jazyce XML

b) WSDL - jazyk pro popis poskytovaných služeb, popisující vlastnosti webové služby a umožňující komunikaci se službou

c) UDDI a WSIL - mechanismy pro nalezení služeb

Přesné vysvětlení webových služeb a jejich principu není jednoduché, proto si zkusíme objasnit především naší problematiku, tj. poslání souboru (obrázku) pomocí webové služby. Pokud tedy budeme chtít poslat nějaký soubor prostřednictvím webové služby, je nutné jej poslat jako pole bajtů v parametru nějaké metody, kterou webová služba nabízí. Soubor by se pak ukládal na disku webového serveru, který webovou službu nabízí. Posílat však velké objemy dat prostřednictvím webových služeb není příliš vhodné. Webové služby k tomu nejsou stavěné. Celý požadavek totiž bude serializovaný a „zabalený“ do SOAP XML zprávy, takže se ve skutečnosti bude přenášet ještě o něco větší datový objem než je skutečná velikost odesílaného souboru. To by mělo velmi nepříznivý vliv na výkon. Webové služby jsou tedy zcela nepoužitelné pro odeslání celého archivu studenta.

5. Vyhodnocení zakázaného chování

Po stažení archivů studentů bude mít vyučující k dispozici podezřelé obrázky, které je možno použít pro celkové vyhodnocení chování studentů. Po dokončení vyhodnocení jedné složky studenta (vyhodnocení lze spustit i pro více studentů najednou) bude systém vždy zobrazovat první ze zachycených obrázků studenta a seznam všech obrázků, ve kterých se vyskytují zakázané ikony. V neposlední řadě bude mít k dispozici také XML protokol. Ten bude sloužit především pro kontrolu obrázků v archivu (celkový počet obrázků, velikosti jednotlivých obrázků) a pro zobrazení některých důležitých informací v něm obsažených. Mezi tyto údaje patří seznam všech procházených internetových stránek, seznam procházených cest souborového systému a seznam procesů.

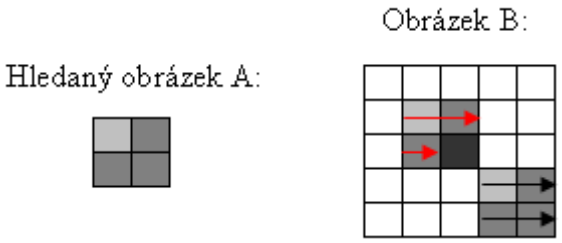
Při vyhledávání systému zakázaných ikon bude v obrázcích vyznačen pouze 100% otisk výskytu ikony a pouze tyto obrázky budou po ukončení vyhledávání zobrazeny. Proto je nutné použít takový formát ukládaných dat, který v rámci komprese nemění tvar ani jas hledaných objektů. V opačném případě je nutno zvýšit toleranci nepodobnosti hledaných vzoru v porovnávaném zdrojovém obrázku. Systém nebude zobrazovat obrázky, u kterých je výskyt zakázané ikony pouze pravděpodobný.

5.1. Hledání ikon v obraze

Existuje mnoho metod jak najít konkrétní obraz (ikonu) v obrázku. V následujícím textu si stručně popíšeme dvě jednoduché metody tohoto vyhledávání.

Hledání konkrétní matice

Všechny rastrové obrázky si můžeme představit jako matici pixelů o různých rozměrech. Při hledání otisku matice obrázku A v matici obrázku B se bude vždy procházet matice B pixel po pixelu (Obrázek 12) a porovnávat s prvním pixelem matice A. V okamžiku shody těchto pixelů se začnou porovnávat ostatní pixely na příslušných souřadnicích. Při prvním neshodném pixelu se další již další pixely nebudou porovnávat. Začnou se dále procházet a porovnávat pixely matice B s prvním pixelem matice A od místa předešlé shody. Při každé další takovéto shodě se začnou opět porovnávat ostatní pixely. Vyhledávání je ukončeno, pokud je dosaženo konce matice B nebo je nalezen 100% otisk matice A. Ukázku tohoto vyhledávání vidíme na následujícím obrázku.

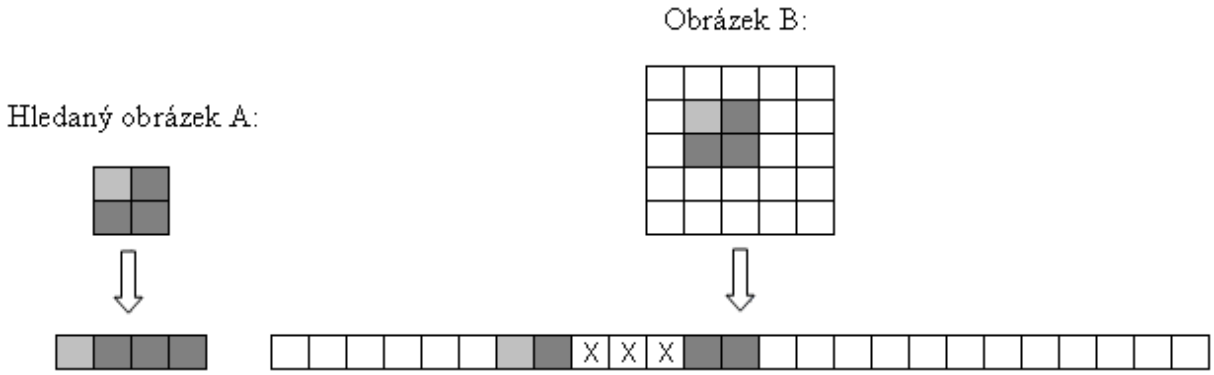


Obrázek 12: Hledání matice

Hledání pomocí rozložení do jednorozměrného pole

Stejně jako v předchozí metodě, hledáme zde shodu prvního pixelu hledaného otisku. Avšak matice jak hledaného obrázku, tak obrázku, ve kterém budeme hledat otisk, budou nejprve rozloženy do jednorozměrného pole. Pole obrázku B se tedy bude procházet a každý jeho pixel porovnávat s prvním pixelem hledaného pole obrázku A. Při shodě se porovná několik dalších pixelů, jejichž počet odpovídá šířce hledaného obrázku - 1. Poté se musí

příslušný počet pixelů pole B přeskočit a až poté porovnávat další pixely obou polí. Opět se při první neshodě začne pole B procházet dále od pozice předešlé shody s prvním pixelem pole A, dokud nedojdeme na konec pole B nebo otisk obrázku A není nalezen. Ukázku tohoto hledání vidíme na obrázku (Obrázek 13).



Obrázek 13: Ukázka rozložení matice, pixely označeny křížkem budou při porovnávání přeskočeny

5.2. Hledání s předzpracováním

Použití předešlých metod k hledání ikon je sice velmi jednoduché, avšak s ohledem na rychlost běhu programu ne příliš vhodné. Při hledání se vždy musí prohledat celý obrázek. Vzhledem k počtu studentů a množství podezřelých obrázků zachycených při jejich testech je hlavní úlohou této metody s předzpracováním zrychlit hledání zakázaných ikon. Základní myšlenka spočívá v předzpracování ikon i obrázků, které je podrobněji rozebráno v následujících kapitolách.

5.2.1. Předzpracování ikon

Předzpracování ikon si rozdělíme na dvě části:

a) Ruční předzpracování
b) Systémové předzpracování

a) Ruční předzpracování

Toto předzpracování spočívá v tom, že všechny ikony, které se považují za zakázané a neměly by se na sejmutých obrázcích vyskytovat, musíme ručně ořezat. Vhodně ořezané ikony pak budou sloužit jako jednotný vzor pro jednoznačné vyhledávání. Stejně jako obrázky sejmuté během testů studentů, je vhodné ukládat ořezané ikony v PNG formátu (viz. kapitola 3.3). PNG formát mimo jiné využívá barevný model RGBA, který vychází z modelu RGB. Model RGBA je rozšířen o tzv. alfa kanál „A“, který nese informaci průhlednosti konkrétního pixelu a může nabývat hodnot 0-255 (pokud je hodnota A=0, pixel je zcela průhledný, pokud A=255, tak pixel není nijak průhledný). Toho je třeba při ořezání ikon využít. Musíme zajistit 100% průhlednost kolem ikon (Obrázek 14 a Obrázek 15) aby bylo možné ikonu jednoznačně vyhledat. Jinak řečeno každý pixel, který přímo nepatří ke konkrétní ikoně, musí mít hodnotu alfa kanálu nastavenou na 0.

Obrázek 14: Neořezaná ikona, 8x přiblíženo

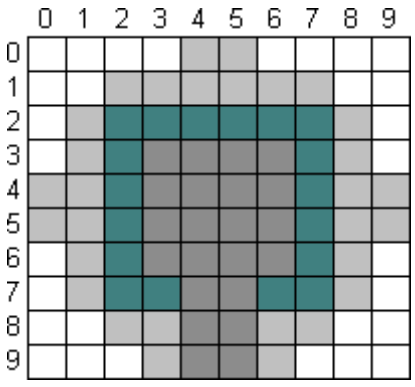
Obrázek 15: Ořezaná ikona, 8x přiblíženo

Velkým problémem jsou ikony s jistou úrovní průhlednosti. Ikony, které mají na nějakou úroveň nastavenou průhlednost jen okraje, je potřeba pouze více ořezat (Obrázek 16). To znamená, že po ořezání zůstanou jen ty pixely, které nejsou nijak průhledné. Ikony, u kterých je alfa kanál „A“ každého pixelu nastaven na jinou hodnotu než 255 nemá smysl ořezávat. Při jiných vzhledech systému Windows či při různých pozadích plochy bude mít takováto ikona vždy jiné složení barev. Pro takové ikony tedy neexistuje jednotný vzor, a nebude je možno jednoznačně vyhledat.

Obrázek 16: Ukázka rozdílů ikon při jiných vzhledech systému Windows

b) Systémové předzpracování

Při tomto předzpracování se využije ručně předzpracovaných ořezaných ikon. Předzpracování bude již realizovat samotný systém. Hlavní myšlenkou tohoto předzpracování je načíst si o každé barvě dané ikony informace, které bude možno při jejím hledání použít. Mějme následující ukázkovou ikonu o rozměrech 10x10px:



Obrázek 17: Příklad ikony, bílé pixely jsou průhledné (složka A=0)

Základní údaj, který je vhodný pro každou barvu ikony uchovávat, je počet pixelů, v kterých se stejná barva v ikoně nachází. To platí i pro pixely s nastaveným alfa kanálem na hodnotu 0. Podobně jako se v kapitole 3.2.3 provádělo hledání nejnižších a nejvyšších souřadnic změn v obrazu, nalezneme nejnižší a nejvyšší souřadnice pro každou barvu dané ikony. Pro pixely s nastaveným alfa kanálem A=0 hledání těchto souřadnic nemá smysl. Tyto pixely nebudou při hledání porovnávány ani jinak používány kromě kontroly jejich počtu v ikoně. Poslední údaj, který se bude pro každou ikonu uchovávat, je pozice prvního neprůhledného pixelu (nebo první pozici pixelu pro každou barvu s alfa kanálem 255). Z ukázkové ikony na Obrázek 17 tedy získáme tyto údaje (Tabulka 4):

Barva	Počet	Horní levý roh	Dolní pravý roh	První
	30	[0, 0]	[9, 9]	4
	18	[2, 2]	[7, 7]	22
	22	[3, 3]	[6, 9]	33
	30	-	-	-

Tabulka 4: Předzpracování ikony

Takováto tabulka bude existovat pro každou zakázanou ikonu a představuje v podstatě jednoduchý hash (otisk) každé ikony. Předzpracování proběhne pro všechny ikony najednou a budeme mít tedy k dispozici seznam takovýchto tabulek.

5.2.2. Předzpracování obrázků

Podobně jako systémové zpracování ikon proběhne i předzpracování samotných obrázků zachycených při testu jednotlivých studentů. Prvním rozdílem je, že předzpracování obrázků daného studenta neproběhne najednou. Každý obrázek se zpracuje právě v okamžiku před samotným hledáním ikon v něm.

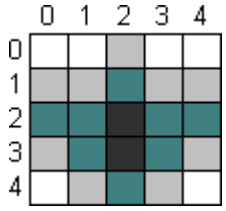
Druhým a zásadním rozdílem je, že se nebudou uchovávat informace o všech barvách obrázku. Pokud by každý obrázek byl sejmutý například na 22“ obrazovce s rozlišením 1920x1080px , mohli bychom dostat seznam řádově až miliónů barev, z nichž naprostá většina by byla zcela irelevantní. Z tohoto důvodu se budou uchovávat informace pouze o těch barvách, které jsou obsaženy v zakázaných ikonách. Jiné barvy a o nich příslušné údaje jsou nepodstatné. Při velkém množství ikon však narůstá i seznam všech barev v nich obsažených. Při procházení obrázku se u každého pixelu musí provést kontrola, zdali je v tomto seznamu jeho barva obsažena, a má tedy smysl zjišťovat informace o dané barvě v obrázku. Operace vyhledání barvy v dlouhém seznamu u každého pixelu obrázku s vysokým rozlišením je časově náročná. Proto je tedy vhodné udržovat zvlášť seznam prvních neprůhledných barev ikony a ke každé položce tohoto seznamu příslušný seznam s ostatními barvami ikony. Při procházení pixelů obrázku se tedy budou nejprve vyhledávat pouze první neprůhledné barvy ikony. Teprve poté, co systém narazí na jednu z nich, smí při průchodu dalšími pixely obrázku vyhledávat v seznamu všech ostatních barev příslušné ikony.

Posledním rozdílem je, že nebudeme potřebovat ani informaci o počtu průhledných pixelů ani o prvních pozicích kterékoliv barvy.

5.2.3. Hledání ikon v obraze s předzpracováním

Předzpracovaného seznamu s informacemi o všech zakázaných ikonách a předzpracovaného obrázku se pak využije při samotném hledání ikon v něm. Seznamem budeme procházet a údaje o barvách každé ikony v něm budeme jistým způsobem porovnávat s údaji o odpovídajících barvách v hashi (otisku) obrázku. Existuje několik situací, kdy lze pomocí předzpracovaných informací takřka ihned jednoznačně určit, zdali se ikona v obrázku může či nemůže nacházet. Pro názornost si nyní ukážeme praktický příklad:

Vezměme si ikonu (Obrázek 18) o rozměrech 5x5 pixelů a k ní příslušné předzpracované údaje (Tabulka 5).

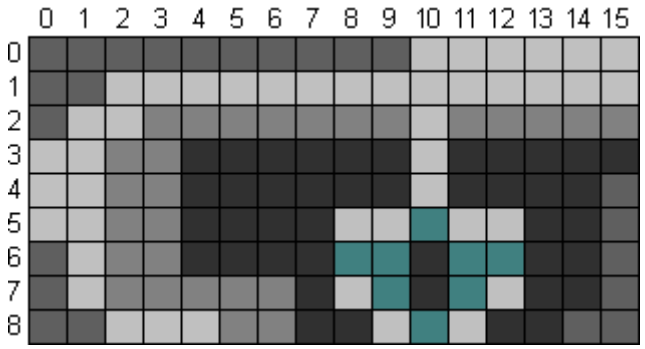


Obrázek 18: Ukázková ikona

Barva	Počet	Horní levý roh	Dolní pravý roh	První
	9	[0, 0]	[4, 4]	2
	8	[0, 1]	[4, 4]	8
	2	[2, 2]	[2, 3]	13
	6	-	-	-

Tabulka 5: Předzpracování ikony

Tuto ikonu se pokusíme nalézt v následujícím obrázku (Obrázek 19) o rozměrech 16x9px předzpracovaným v pod ním uvedené tabulce (Tabulka 6).



Obrázek 19: Ukázkový obrázek pro vyhledávání

Barva	Počet	Horní levý roh	Dolní pravý roh	První
	44	[0, 0]	[15, 8]	2
	42	[4, 3]	[15, 8]	13
	8	[8, 5]	[12, 8]	8

Tabulka 6: Předzpracování ukázkového obrázku

Barvy ikony s příslušnými informacemi se budou postupně procházet od první barvy v seznamu a budou se konfrontovat s informacemi téže barvy obrázku. Pomocí těchto porovnávání však nelze jednoznačně určit, zdali se ikona v obrázku nachází. Což v konečném důsledku znamená, že pokud chceme jednoznačně určit výskyt ikony v obrázku, nelze se vyhnout přesnému dohledání konkrétní ikony pomocí jedné z metod uvedených v kapitole 5.1. Při použití těchto metod se však nebudou porovnávat pixely s nastaveným alfa kanálem na hodnotu 0 (nebo A < 255). Je potřeba jen spočítat počet neporovnaných pixelů při hledání, a pokud všechny ostatní neprůhledné barvy odpovídají, tento počet porovnáme s počtem průhledných pixelů v ikoně. Aby došlo ke zrychlení tohoto dohledání, je třeba z informací o barvách určit a přepočítávat meze (souřadnice), mezi kterýma bude dohledání probíhat.

Ukázkový postup při hledání ikony s předzpracováním:

a) Nejprve zjistíme, zdali se každá barva ikony v obrázku vůbec nachází. Náš ukázkový obrázek obsahuje všechny barvy ikony a má tedy smysl pokračovat v hledání.

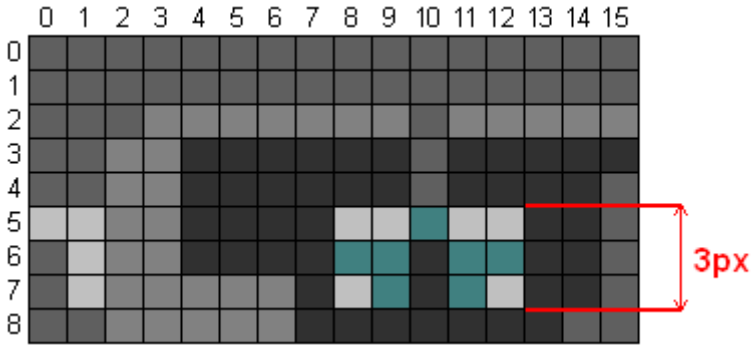
b) U každé barvy ikony zjistíme, zdali je počet stejně obarvených pixelů menší či roven počtu pixelů téže barvy v obrázku. Pokud je počet pixelů dané barvy v obrázku menší, není potřeba ikonu dále vyhledávat. V našem příkladu má ikona první barvu v 9 pixelech a obrázek ve 44 pixelech, počty ostatních barev jsou také v požadované četnosti.

c) Zjistíme, zdali se výskyt každé barvy nenachází v obrázku v užší oblasti než v ikoně. Pokud odečteme horní souřadnice ikony od dolních souřadnic konkrétní barvy, musí být x-ová i y-ová hodnota výsledných souřadnic menší nebo rovna příslušným souřadnicím rozdílu souřadnic téže barvy v obrázku. Tento krok si pro náš příklad znázorníme následující tabulkou (Tabulka 7).

Barva	Velikost oblasti výskytu barvy v ikoně	Velikost oblasti výskytu barvy v obrázku	Porovnání výsledných souřadnic: v ikoně <= v obrázku
	[4, 4] - [0, 0] = [4, 4]	[15, 8] - [0, 0] = [15, 8]	OK
	[4, 4] - [0, 1] = [4, 3]	[12, 8] - [8, 5] = [4, 3]	OK
	[2, 3] - [2, 2] = [0, 1]	[15, 8] - [4, 3] = [11, 5]	OK

Tabulka 7: Znázornění porovnání šířky oblasti výskytu barev

Z tabulky vyplývá, že pro náš ukázkový příklad šířky oblastí pro každou barvu vyhovují. V následujícím obrázku a tabulce (Obrázek 20, Tabulka 8) je možné vidět příklad, ve kterém se ukázková ikona z Obrázek 18 dle prvních dvou kroků postupu vyhledávání ikony (každá barva ikony se v obrázku nachází a souhlasí počty všech barev) v obrázku ještě může vyskytovat. Avšak šířka oblasti hned první barvy ikony v obrázku již nevyhovuje a nemá tedy smysl ikonu dále vyhledávat.



Obrázek 20: Ukázka obrázku s nevyhovující šířkou oblastí první barvy ukázkové ikony, která musí být alespoň 5px dle ikony

Barva	Velikost oblasti výskytu barvy v ikoně	Velikost oblasti výskytu barvy v obrázku	Porovnání výsledných souřadnic: v ikoně <= v obrázku
	[4, 4] - [0, 0] = [4, **4**]	[12, 7] - [0, 5] = [12,**2**]	Nevyhovuje

Tabulka 8: Nevyhovující y-ová hodnota souřadnice

d) Začneme uchovávat meze, které udávají oblast, ve které se ikona může nacházet. Kdekoliv mimo meze se již ikona pro případné konečné dohledání nemůže vyskytovat. Tyto meze budou nejprve nastaveny při procházení první barvy ikony. První barva ikony v našem obrázku má levý horní roh výskytu na souřadnicích [0,0], tyto souřadnice budou nastaveny jako levý horní roh mezí. Pravý dolní roh mezí získáme připočtením rozměrů ikony (zmenšeným o jedna) k souřadnicím posledního výskytu první barvy ikony v obrázku takto: [4, 4] + [15, 8] = [19, 12]. Pokud meze přesahují rozměry obrázku tak, jak je tomu v tomto případě, je vhodné je aktualizovat na maximální hodnoty souřadnic obrázku. Meze tedy budou nastaveny na hodnoty [0, 0] a [15, 8]. Pro všechny další barvy lze tyto meze zmenšit. To ale pouze v případech, kdy pro další barvu platí, že její horní souřadnice jsou větší či dolní souřadnice jsou menší než souřadnice mezí. Jsou-li meze nastaveny na [0, 0] a [15, 8] a další barva ikony se v obrázku vyskytuje pouze mezi souřadnicemi [8, 5] a [12, 8], je možné, že oblast mezí lze ještě zmenšit a tím i oblast pro případné dohledání ikony. Následuje celý výpočet mezí pro náš příklad:

- Meze tedy budou nejprve inicializovány podle souřadnic první barvy ikony v obrázku na [0, 0] a [15, 8].

- Druhá barva ikony se v obrázku nachází mezi souřadnicemi [8, 5] a [12, 8] má tedy smysl zkusit vypočítat nové meze:

  Horní levé souřadnice mezí vypočteme: $[8,5]-([0,1]-[0,0])=[8,4]$
  - kde [0, 1] a [0, 0] jsou souřadnice horního rohu druhé a první barvy ikony

  Dolní pravé souřadnice mezí vypočteme: $[12,8]+([4,4]-[4,4])+[0,0]=[12,8]$
  - kde souřadnice [4, 4] jsou souřadnice dolního rohu druhé a první barvy ikony a [0, 0] je x-ová a y-ová hodnota vzdálenosti posledního výskytu druhé barvy ikony od pravého a dolního okraje ikony

Obě souřadnice nově vypočtených mezí vymezují menší oblast než dosud nastavené meze (původní horní levé souřadnice jsou menší a dolní pravé souřadnice jsou větší). Proto tyto vypočtené meze označíme za aktuální. Předešlé meze již nebudeme potřebovat. Obecné vzorce pro výpočet horních a dolních mezí budou vypadat takto (Rovnice 1, Rovnice 2):

$$M_H = O_{AH} - (I_{AH} - I_{1H})$$

Rovnice 1: Výpočet souřadnic horního levého rohu mezí

$$M_D = O_{AD} + (I_{AD} - I_{1D}) + V_{AD}$$

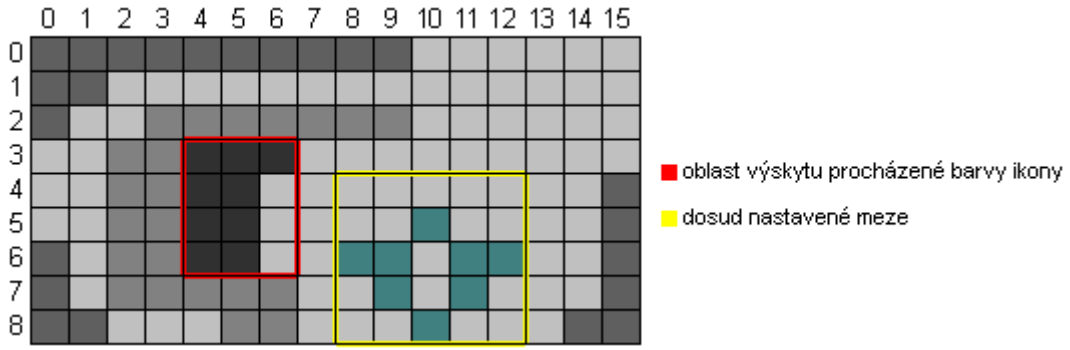
Rovnice 2: Výpočet souřadnic dolního pravého rohu mezí

M_H - Výsledné souřadnice horního pravého rohu mezí
O_{AH} - Horní souřadnice aktuální barvy v obrázku
I_{AH} - Horní souřadnice aktuální barvy v ikoně
I_{1H} - Horní souřadnice první barvy ikony

M_D - Výsledné souřadnice dolního levého rohu mezí
O_{AD} - Dolní souřadnice aktuální barvy v obrázku
I_{AD} - Dolní souřadnice aktuální barvy v ikoně
I_{1D} - Dolní souřadnice první barvy ikony
V_{AD} – Vzdálenost dolních souřadnic aktuální barvy od okraje ikony

- Třetí barva naší ukázkové ikony se nachází mezi souřadnicemi [4, 3] a [15, 8]. Oblast mezí nemůže být menší a nemá smysl provádět další výpočet.

e) Zjistíme, zdali se některá právě procházená barva nenachází pouze v oblastech mimo oblast dosud nastavených mezí. Jinými slovy řečeno, oblast výskytu právě procházené barvy musí protínat oblast mezí. Ukázkový příklad splňuje i tuto podmínku a je tedy možné přejít k poslednímu bodu postupu vyhledávání. Na následujícím obrázku (Obrázek 21) si však ukážeme příklad, kdy po tomto kroku již nemá smysl pokračovat. Jak bylo uvedeno v předchozím kroku postupu vyhledávání (ad. d)), budou i zde souřadnice mezí po průchodu druhé barvy ikony nastaveny na [8, 4] a [12, 8]. Třetí barva ukázkové ikony se ovšem nachází pouze mezi souřadnicemi [4, 3] a [6, 6] (červeně vyznačeno v obrázku) a ikona se v obrázku tedy již nemůže vyskytovat.



Obrázek 21: Ukázka obrázku, kde se oblast výskytu právě procházené barvy nachází mimo oblast dosud nastavených mezí

f) Pokud jsou splněny všechny výše uvedené podmínky, je velmi pravděpodobné že ikona se v obrázku může nacházet. V našem příkladu tomu tak je, avšak nelze určit 100% výskyt ikony. Jak již bylo řečeno, je nutné při použití této metody konečné dohledání ikony.

5.3. Porovnání

Porovnání obyčejného vyhledávání ikony s vyhledáváním s předzpracováním vidíme v níže uvedené tabulce (Tabulka 9). Pro porovnávání byl použit jediný náhodný obrázek, na kterém se vyskytovalo 1 až 5 ikon, z nichž 3 měly první neprůhlednou barvu totožnou. Všechny ikony byly rozmístěny v dolní polovině obrázku. Postupně byla hledána jedna až všech pět testovacích ikon. Samotné předzpracování ikon proběhne ihned po výběru složky se zakázanými ikonami. Dobu tohoto předzpracování můžeme zanedbat, protože proběhne před samotným spuštěním vyhledávání a nejedná se o časově náročnou operaci.

Hledaných ikon	Nestejných ikon v obrázku	Doba vyhledávání bez předzpracování [ms]	Předzpracování obrázku [ms]	Celková doba vyhledávání s předzpracováním obrázku [ms]
1	1	2414	1031	1065
2	1	3611	1099	1115
3	1	5426	1126	1167
4	1	7136	1168	1198
5	1	8816	1202	1255
2	2	3633	2681	2739
3	2	5449	2749	2800
4	2	7166	2790	2815
5	2	9157	2908	2945
3	3	5450	2755	2768
4	3	7286	2819	2826
5	3	9265	2827	2887
4	4	7298	2750	2939
5	4	9307	2941	2956
5	5	9165	3020	3101

Tabulka 9: Porovnání vyhledávání ikon s předzpracováním

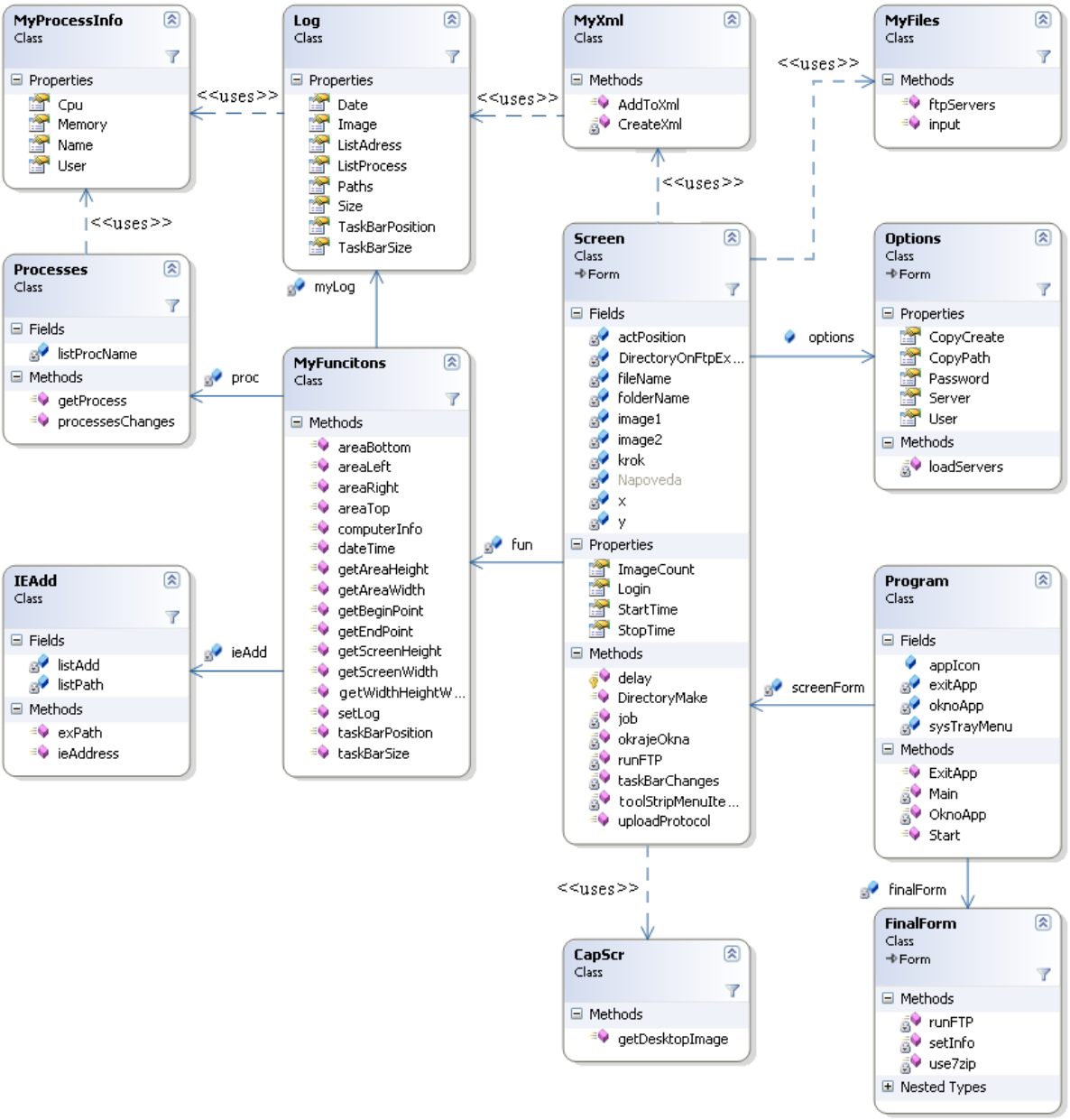
Z tabulky vyplývá, že efektivita rychlosti vyhledávání s předzpracováním roste s počtem hledaných ikon v obrázku. Dále můžeme vidět, že doba dohledávání ikon je zanedbatelná oproti době předzpracování samotného obrázku. To je především důsledkem správného nastavování mezí pro ikony a také toho, že je velmi nepravděpodobné, že barvy jednotlivých ikon (především první neprůhledné barvy) se v obrázku vyskytují i v jiných oblastech mimo případný výskyt zakázané ikony.

6. Návrh implementace

Hlavní částí této bakalářské práce je navrhnout a implementovat dvě aplikace, pomocí nichž lze vyhodnotit chování studenta v průběhu online testů. Předešlé kapitoly se zabývaly analýzou problému a některými zásadními metodami, které lze při implementaci použít. V následujících kapitolách je uvedena struktura aplikací pro jejich následnou implementaci pomocí UML diagramů tříd a také návrh uživatelského rozhraní některých formulářů. Jak již bylo řečeno v úvodu, aplikace budou vyvíjeny v prostředí vývojového nástroje Microsoft Visual Studio 2008 a zdrojové kódy budou psány v jazyce Csharp.

6.1. Aplikace „Hlídání obrazovky“

Třídní diagram:



Seznam tříd (popis, charakteristika):

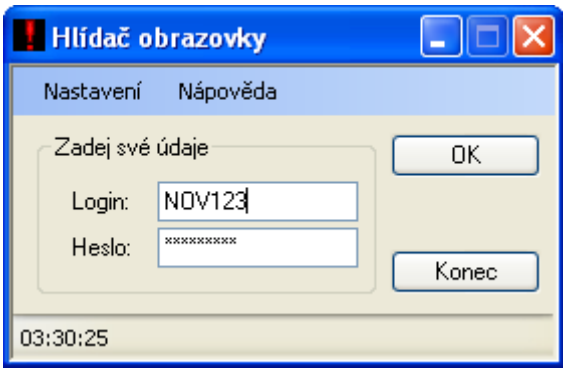
Program:

Třída obsahující metodu main(). Dále se v této třídě načte tray-ikona celé aplikace, její nabídka a k ní příslušné akce jako je například ukončení hlídání obrazovky.

Screen:

Formulář pro zadání loginu a hesla studenta. Po potvrzení bude formulář skryt a začne probíhat hlídání obrazovky. Návrh GUI formuláře vidíme na obrázku níže (Obrázek 22). Třída bude obsahovat tyto základní proměnné (či properties) a funkce:

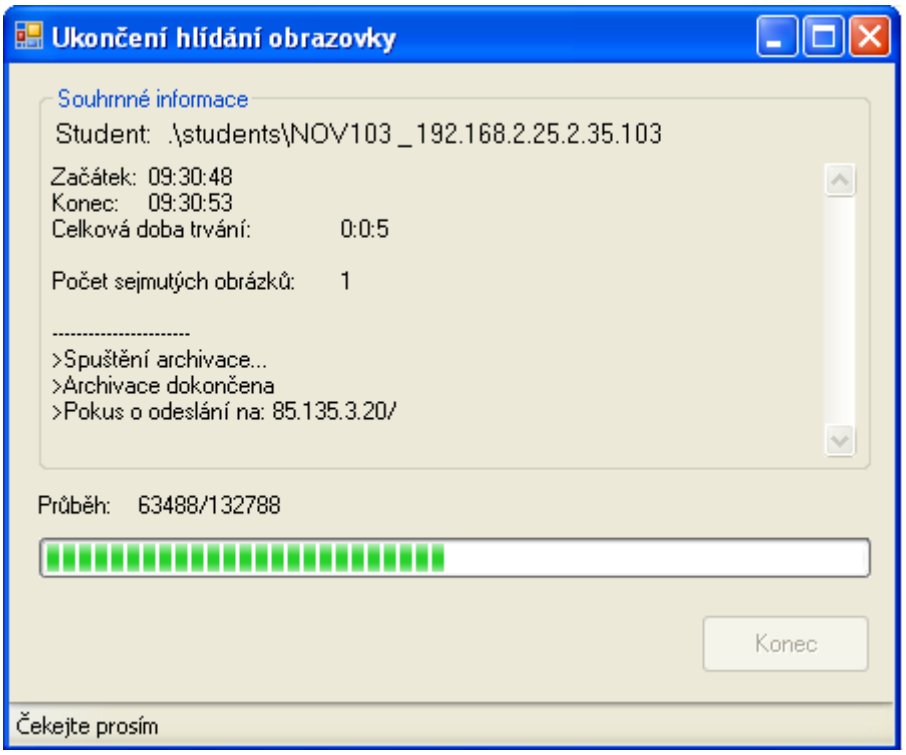
- image1 a image2 – bitmapy dvou právě porovnávaných obrázků
- actPosition – aktuální pozice hlavního panelu
- options –instance třídy Options pro nastavení aplikace
- fun – instance třídy Functions pro pomocné funkce
- ImageCount – počítadlo uložených obrázků
- Login – identifikace studenta
- StartTime a StopTime – čas začátku a konce hlídání obrazovky
- winMargins() – funkce vracející hodnotu true, po nalezení okraje okna (viz. kapitola 3.2.3).
- taskBarChanges() – funkce vracející hodnotu true, při nalezení změn v hlavním panelu.
- delay() – hlavní funkce programu pro řízení sejmutí obrazovky a případného uložení obrázku. Bude využívat privátní funkce winMargins() a taskBarChanges().
- directoryMake() – funkce pro vytvoření adresáře studenta na FTP serveru.
- runFTP() – funkce pro průběžné odeslání jednoho souboru či obrázku na FTP server.
- serviceWork() – funkce pro průběžné odeslání obrázku pomocí webových služeb.



Obrázek 22: Návrh uživatelského rozhraní formuláře Screen

FinalForm:

Formulář, který se zobrazí po ukončení hlídání obrazovky. Zobrazí studentovi informace o průběhu hlídání, jako je celkový počet sejmutých obrázků a dobu hlídání obrazovky. Mimo jiné zde bude dále zobrazen celkový průběh případného odesílání archivu na FTP server. Obsahuje funkce pro archivaci složky studenta (use7zip()) a pro odeslání celého archivu na FTP server (runFTP). Návrh uživatelského rozhraní tohoto formuláře je na následujícím obrázku (Obrázek 23):



Obrázek 23: Návrh uživatelského rozhraní formuláře FinalForm

Options:

Formulář pro nastavení aplikace. Tj. především nastavení FTP serveru pro odesílání zachycených dat.

MyFunctions:

Třída bude obsahovat všechny potřebné pomocné funkce aplikace. Mezi ně patří např. funkce pro zjištění IP adresy a názvu počítače, na kterém je aplikace spuštěna (computerInfo()), funkce pro výpočet velikosti hlavního panelu (taskBarSize()), funkce pro určení pozice hlavního panelu (takbarPosition()) a v neposlední řadě také funkce pro nastavení záznamu určeného pro uložení do protokolu (funkce setLog(), viz. kapitola 3.4).

MyFiles:

Třída pro práci se vstupními soubory. Obsahuje funkce pro načtení možných adres FTP serverů a pro načtení konfigurace aplikace ze souboru (čas mezi sejmutím obrazovky a procentuálně zadaná mez možných změn v obraze).

Log:

Třída reprezentující jeden záznam v XML protokolu k jednomu obrázku. Tj. Název obrázku a k němu příslušné informace o hlavním panelu, velikosti obrázku, data uložení, seznamu procesů, seznamů cest a adres.

MyXml:

Třída pro práci s XML protokolem. Vytvoření XML souboru protokolu, zápis jednoho záznamu typu Log do protokolu.

CapScr:

Třída nabízející funkci pro sejmutí obrazovky.

IEAdd:

Třída obsahující funkce pro zjištění dosud nezaznamenaných adres v prohlížeči Internet Explorer a cest navštívených v souborovém systému.

Processes:

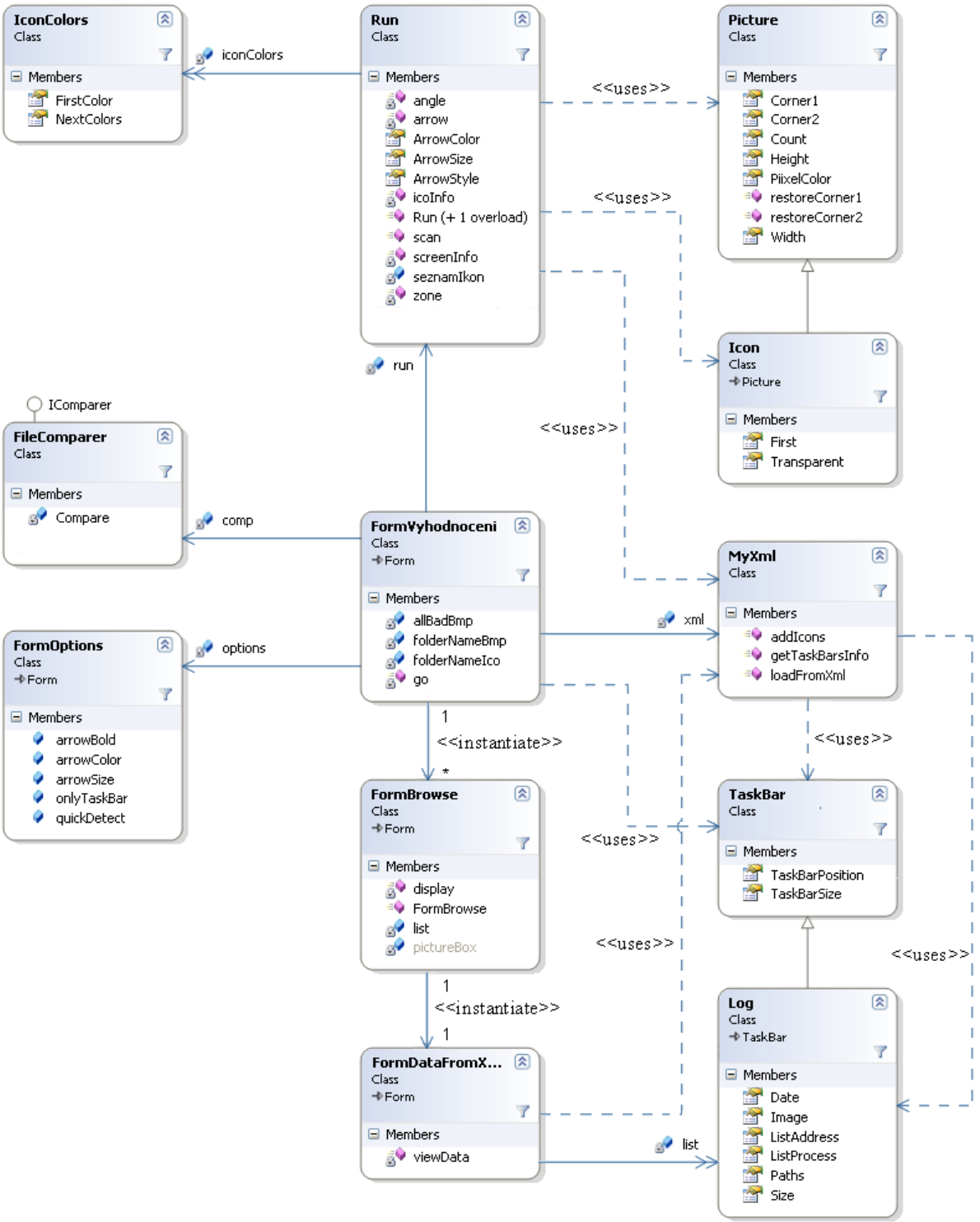
Třída obsahující funkce pro zjištění změn v procesech (viz. kapitola 3.1) a pro zjištění dosud nespuštěných procesů během spuštěného hlídání.

MyProcessInfo:

Třída reprezentující záznam informací jednoho procesu.

6.2. *Aplikace „Vyhodnocení“*

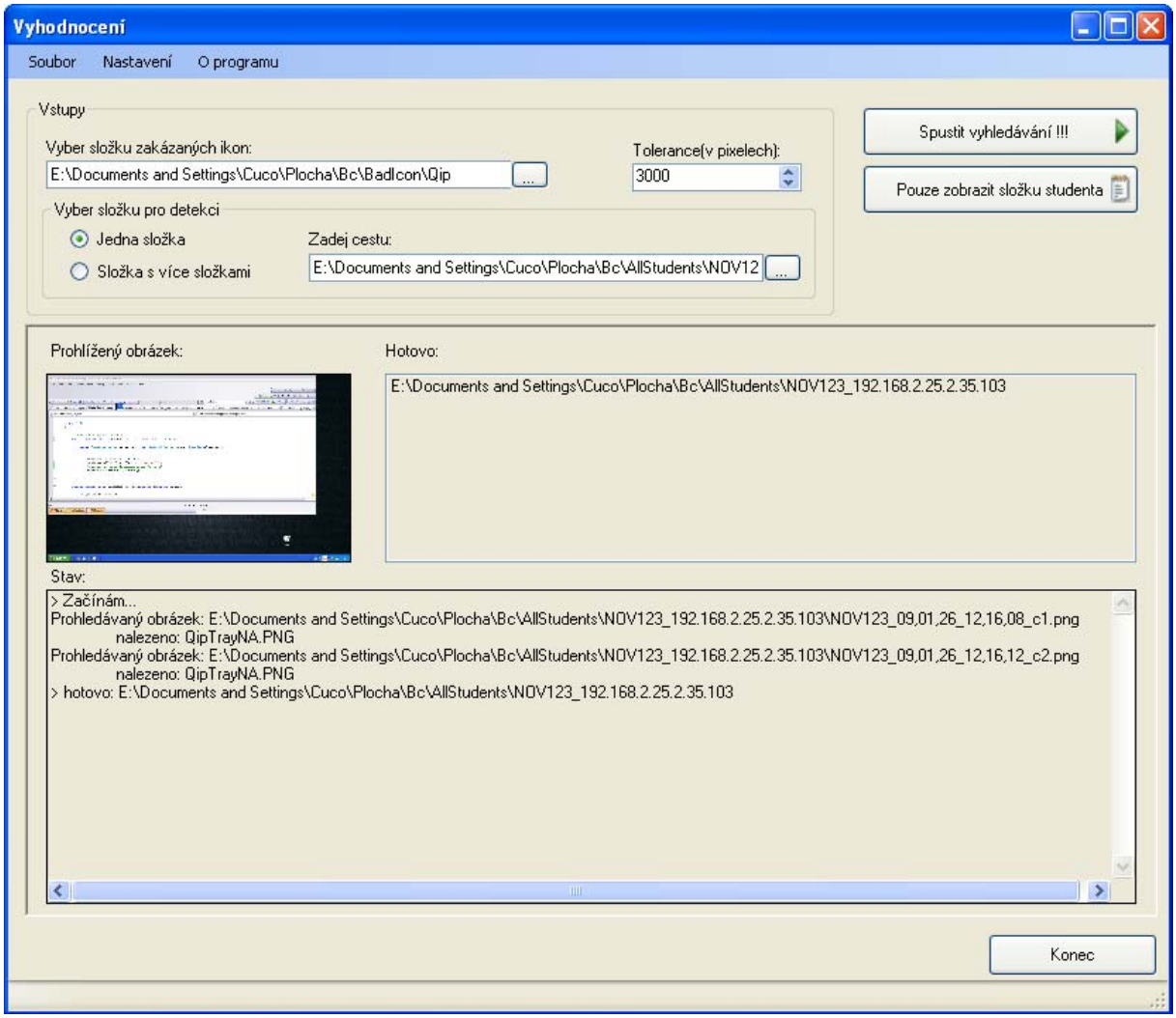
Třídní diagram:



Seznam tříd (popis, charakteristika):

FormVyhodnocení:

Úvodní formulář, ve kterém se po zadání složky studentů (folderNameBmp) s jejich zachycenými daty spustí vyhledávání zakázaných ikon z vybrané složky (folderNameIco). To bude provedeno pomocí metody hlavní metody go() spuštěné v novém vlákně. Dále bude obsahovat instanci třídy options pro možná nastavení vyhledávání a instanci třídy Comparer, implementující rozhraní IComparer, k účelu načtení obrázků setříděných podle jména. Na následujícím obrázku (Obrázek 24) je návrh uživatelského rozhraní tohoto formuláře.



Obrázek 24: Návrh GUI formuláře "Vyhodnocení"

FormBrowse:

Formulář, který bude možno otevřít po dokončení vyhledávání v obrázcích jednoho studenta. Bude sloužit k zobrazení (metoda display ()) jeho zachycených a vyhodnocených obrázků (v seznamu "list"). Pokud se v obrázcích budou nacházet zakázané ikony, budou již vyznačeny. Zobrazení jednotlivých obrázku ze seznamu vyhodnocených umožní komponenta formuláře pictureBox.

FormDataFromXml:

Formulář pro zobrazení podrobností z XML protokolu studenta (funkce viewData()). Kromě zachycených navštívených adres, cest souborového systému a procesů bude zobrazovat názvy obrázků, jejichž velikost nesouhlasí s velkostí uvedenou v protokolu nebo jejichž název není v protokolu uveden.

FormOptions:

Formulář pro nastavení vyhledávání a označení případných nalezených ikon. Základní tributy třídy:

- quickDetect - nastavení, které udává, že po nalezení první libovolné ikony ihned přejdeme na další obrázek.
- onlyTaskBar – udává možnost vyhledávání ikon pouze v pravděpodobné oblasti hlavního panelu načtené z protokolu.
- arrowBold, arrowSize, arrowColor – nastavení stylu, velikosti a barvy pro šipku ukazující na nalezenou ikonu.

FileComparer:

Třída sloužící pro setřídění obrázků podle jejich názvu (metoda compare()).

MyXml:

Třída pro práci s XML protokolem obsahující tyto základní funkce:

- loadFromXml() – funkce vracející seznam všech záznamů z protokolu.
- addIcons() – funkce pro zápis nalezených ikon do protokolu k příslušnému obrázku
- getTaskBarInfo() – funkce vracející informace o hlavním panelu pro příslušné obrázky

Picture a Icon:

Třídy reprezentující informace pro předzpracování ikon a obrázků viz. kapitola 5.2.

Log:

Třída reprezentující jeden záznam v protokolu.

Run:

Třída obsahující nejdůležitější funkce aplikace. Funkci pro předzpracování ikon (doIcoInfo()), funkci pro předzpracování obrázků studenta (doScreenInfo()) a funkci pro samotné hledání ikon (scan()), vracející seznam obrázků s vyznačenými zakázanými ikonami. Dále obsahuje privátní funkce pro vyznačení oblasti zakázané ikony a vykreslení šipky k ní do obrázku (zone(), arrow()).

7. Experimentální testování

V této kapitole se budeme zabývat testováním jednotlivých aplikací po jejich implementaci (základních funkcí).

7.1. Aplikace „Hlídání obrazovky“

Testování aplikace „hlídání obrazovky“ bude probíhat pomocí nástroje pro snímání pracovní plochy [1], který vytvoří video soubor, na němž bude zachyceno veškeré dění na obrazovce. Aby bylo možné z videa snímat obrázky, je nutné použít nástroj, ze kterého lze obrázky snímat během přehrávání videa [2]. Pro testování postačí 5 minutové video, na kterém je zachycen průběh zkušebního online testu během kterého jsou prováděny zakázané aktivity jako je otevření souboru z flash disku, ICQ připojení, použití Total commanderu, spuštění a zadávání adres v internetovém prohlížeči. Při snímání tohoto videa však není možné kontrolovat pozici či změny v hlavním panelu (pozici či velikost hlavního panelu z videa nelze zjistit ze systémových informací). Nelze sledovat ani změny v procesech či využití procesoru a je tedy možné využít pouze hledání změn na pracovní ploše zachycené ve videu.

Stejně tak nelze při využití tohoto testování vyzkoušet přístupu hlídání pomocí spuštěných procesů či využití paměti.

- Základní testovací časový interval mezi sejmutím obrazovky byl nastaven na 2 sekundy. Pokud by během pěti minut byla obrazovka každé 2 vteřiny sejmuta a obrázek byl ihned uložen, je snadné si dopočítat, že bychom dostali celkem 150 obrázků. Ve skutečnosti je nutné k nastaveným 2 vteřinám připočíst čas potřebný k sejmutí obrazovky, uložení obrázku a případné další operace. Při tomto testu jsme dostali 129 obrázků. Na obrázcích nechybí zachycená žádná podstatná aktivita, avšak naprostá většina obrázků je zcela irelevantních. Celá složka má velikost 12,2 MB.
- Aplikace „hlídání studenta“ za použití metod hledání změn v obraze zachytila při přehrávání testovacího videa při 2 vteřinových intervalech mezi sejmutím obrazovky 59 obrázků. Mimo tento interval byl nastavený krok průchodu obrázku na hodnotu 4 (viz. kapitola 3.2.1), dále nastavená hranice možných změn v obraze na 10% a počet neshodných pixelů určující možný výskyt nového okraje okna na 10 % výšky obrazovky (viz. kapitola 3.2.3). Všechny podstatné a důležité zakázané aktivity jsou mezi obrázky zachyceny. Složka s obrázky měla velikost 5,90 MB.
- Po intervalu nastaveném na 4 vteřiny bylo naší aplikací zachyceno jen 42 obrázků, jejichž celková velikost včetně protokolu byla 3,93 MB. Opět v obrázcích nechyběly žádné zakázané aktivity z videa.
- Pokud byl interval nastavený na časový úsek delší než 4 vteřiny, stávalo se, že mezi obrázky již chyběl alespoň jeden důležitý pro vyhodnocení. Z tohoto důvodu není vhodné nastavovat interval sejmutí obrazovky delší než právě 4 vteřiny.

Za použití průběžného odesílání na FTP server nedocházelo k žádným problémům a všechny obrázky byly vždy včetně protokolu odeslány, i když několik posledních až po

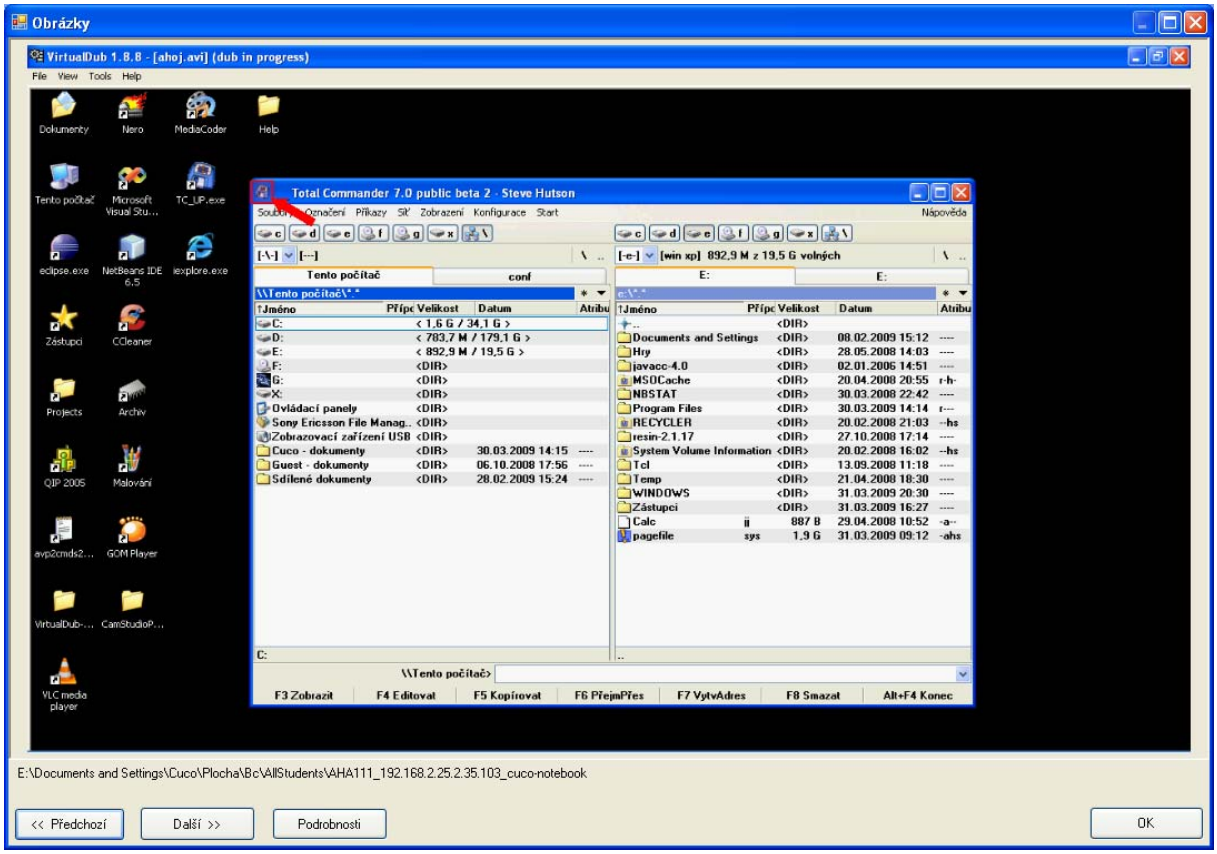
[1] Freeware nástroj CamStudio Portable 1.2.2.0
[2] Freeware nástroj VirtualDub 1.8.8

ukončení hlídání. Bohužel však byly obrázky posílány pouze z jednoho počítače. Při použití možnosti archivace složky a následného odeslání celého archivu po ukončení hlídání, se projevila doba potřebná k archivaci a odeslání jako velká nevýhoda.

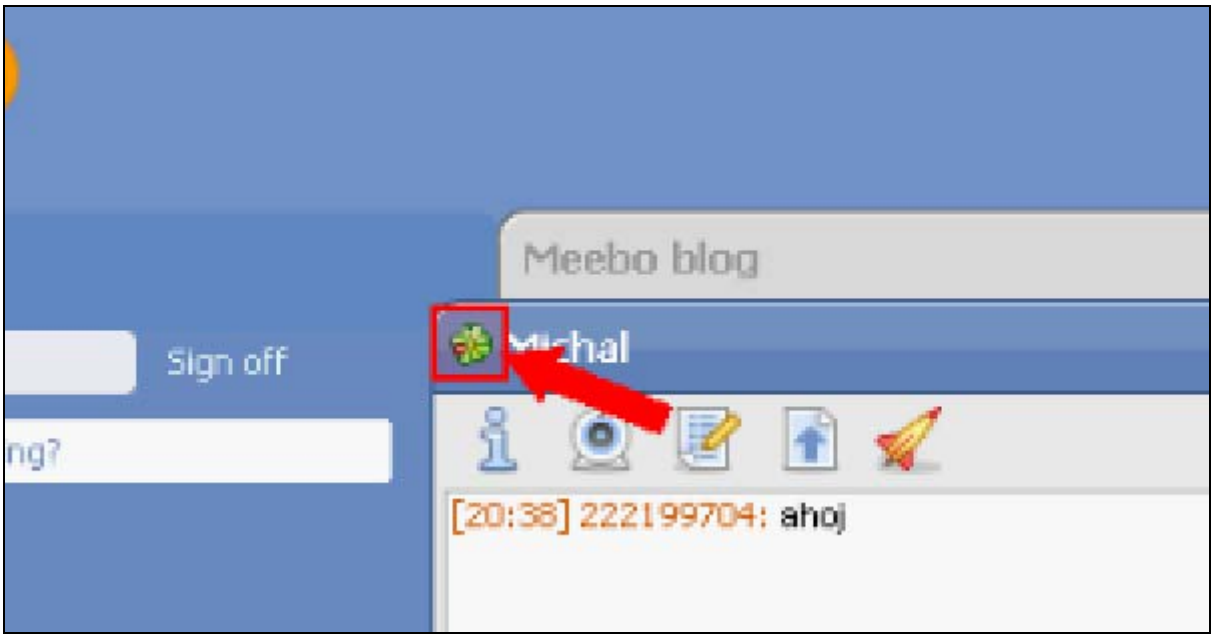
7.2. Aplikace „Vyhodnocení“

Pro test aplikace „Vyhodnocení“, za použití metody vyhledávání z předzpracováním uvedené v kapitole 5.2, bylo následně použito 42 obrázků zachycených po intervalu sejmutí 4 sekundy. Test proběhl se sadou 40 zkušebních ikon, z nichž dvě se v obrázcích vyskytovaly alespoň jednou. Samotné hledání ikon ve všech obrázcích trvalo přibližně 2 minuty a 51 vteřin

Na následujících obrázcích jsou ukázky zobrazení podezřelých obrázků v aplikaci „Vyhodnocení“ s nalezenými ikonami:



Obrázek 25: Ukázka zachyceného obrázku s nalezenou ikonou total commanderu



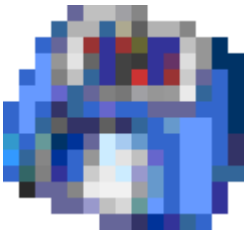
Obrázek 26: Vyznačení nalezené zakázané ikony (4x přiblíženo)

8. Možné optimalizace

8.1. Optimalizace přenosu

Optimalizace přenosu zachycených dat spočívá v možnosti odesílání menšího datového objemu. Námi použité metody vyhledávání v obrazu pracují s obrázky v barevném prostoru RGB, kde výsledná barva jednotlivých pixelů obrázku je dána složením tří barev – Red, Green, Blue (červená, zelená, modrá). Každá z těchto barevných složek může nabývat hodnot 0-255. Tento formát se nazývá „TrueColor“ neboli věrné zobrazení barev. Zároveň jsou tyto obrázky bezeztrátově komprimované a je tedy možno v nich vyhledat 100% otisk zakázané ikony. Bezeztrátové metody komprese zdaleka nedokážou snížit datový objem obrázků tak jako metody ztrátové s vyššími úrovněmi komprese. Pro optimalizaci přenosu lze použít například tyto přístupy:

Odesílání JPEG - Nejpoužívanější formát využívající ztrátové komprese je JPEG. Při přenášení obrázků v tomto formátu by jistě byl odesílaný nejmenší možný datový objem. Již při 50% úrovni komprese je patrná ztráta kvality obrazu jak můžeme vidět na následujících obrázcích (Obrázek 27 a Obrázek 28).



Obrázek 27: Ikona komprimovaná bezeztrátově 8x přiblíženo

Obrázek 28: Ikona - 50% komprese JPEG 8x přiblíženo

Je tedy patrné, že při vyhledávání ikony v JPEG obrázku není možné jednoznačně určit, zdali se v něm ikona nachází. Kromě ztráty informací o některých barvách JPEG komprese může deformovat i tvary. Další podstatnou nevýhodou je, že JPEG nepodporuje průhlednost. Pokud by tedy byly i ikony ukládány v JPEG formátu, musel by se vždy vyhledávat obdélníkový otisk ikony. Případně by se pak ty barvy pixelů ikony, které by ve skutečnosti měli být průhledné a neměli by se porovnávat, museli být nahrazeny jinou barvou. Tato barva by měla být vybrána z barev, které jsou nejméně používané v počítačové grafice.

Převod do šedé škály - Při převodu TrueColor obrázků do šedé škály lze získat nejvíce 256 odstínů šedi. To vede k celkovému podstatnému zmenšení velikosti obrázku. Lze toho jednoduše docílit pomocí následujícího vzorce:

$$I = r \cdot 0{,}229 + g \cdot 0.587 + b \cdot 0{,}114$$

Rovnice 3: Výpočet intenzity bílé při převodu do GrayScale

Ve vztahu (Rovnice 3: Výpočet intenzity bílé při převodu do GrayScale) jsou proměnné r, g, b složky vstupní barvy a I je výsledná intenzita bílé [4]. Je tedy jasné, že výsledná intenzita bílé bude stejná pro několik kombinací barevných složek. Jinými slovy řečeno, z více různých barev můžeme dostat jeden a tentýž odstín šedi, čímž se ztrácí možnost jednoznačného dohledání. Ikony pro vyhledávání by v tomto případě museli být také převedeny do šedé škály. Z rovnice 3 vyplývá, že výsledná intenzita bílé nepočítá se složkou „A“ nesoucí informaci o průhlednosti pixelu. Stejně jako v předchozím případě zde tedy nastává problém s průhledností.

Odesílání GIF - Podobně jako lze získat pouze 256 odstínů šedi, lze obrázek uložit ve formátu GIF, který umožňuje uložit obrázek pouze ve 256 různých barvách (8 bitů) z libovolně definované palety. Navíc jedna položka z této přiřazené palety smí být označena jako zcela průhledná [3]. To by mělo opět za následek podstatné zmenšení velikosti obrázku. Tato paleta 256 barev by musela být vhodně zvolena dle možné ikony (ikon) nacházející se v obrázku, aby bylo při vyhodnocení možné její co nejpřesnější dohledání

**Poznámka*: Při všech těchto výše zmíněných přístupech optimalizace nelze dohledat 100% otisk konkrétní ikony. Z těchto důvodů by se pro vyhledávání na vstupu procentuálně zadávalo, jaká je možnost výskytu ikony v obrázku.

Odesílání částí obrázku – Jednou z dalších možností je zasílat jen části obrázku. Tyto části by byly komprimovány bezeztrátově a jednalo by se o tu část obrázku, odpovídající oblasti mezi první a poslední změnou obrázku s předešlým. Pokud bychom chtěli mít však obrázek úplný ve skutečných rozměrech, bylo by možné všechny ostatní oblasti uložit komprimované ztrátově. Při vyhodnocení by pak byly všechny části zobrazeny sloučené a v nekomprimované oblasti by bylo možné dohledat 100% otisk konkrétní ikony.

8.2. Ostatní alternativy

Částečné vyhodnocení na straně studenta – počet odesílaných obrázků lze jednoduše omezit jejich vyhodnocením přímo v klientské aplikaci, tzn. při průběhu probíhajícího testu studenta. Na centrální úložiště by pak byly odesílány jen obrázky s výskytem alespoň jedné ze zakázaných ikon a také obrázky, u kterých bylo zjištěno velmi výrazných změn v obraze. Výhodou tohoto přístupu je samozřejmě menší přenos dat ovšem na úkor velkého zatížení počítače na straně studenta provádějícího online test.

Celkové vyhodnocení na straně studenta (tzv. „security gate“) – při tomto přístupu by celá aplikace hlídání obrazovky včetně celkového vyhodnocení byla jako jeden celek spuštěna na klientské straně. Nedocházelo by tak k žádnému odesílání dat na vzdálené úložiště. Při zjištění provádění jakékoliv nekorektní práce (při nalezení zakázané ikony) během právě probíhajícího testu by došlo například k zablokování probíhajícího testu či celého počítače. To až do doby zadání správného hesla, které dovolí pokračování v testu. Toto heslo by mohl sdělovat dohlížející vyučující, který by tak již věděl, že daný student prováděl nějakou nedovolenou aktivitu. Velkou nevýhodou je opět zatížení počítače, na kterém je prováděn online test. K sejmutí obrazovky by nemělo docházet dříve, než bude ukončeno vyhledávání ikon v předchozím obrázku. Poněvadž doba potřebná pro vyhledávání ikon není pro všechny obrázky konstantní (řádově může být pro některé obrázky různá o několik vteřin), bylo by vhodné vytvořit v takovéto aplikaci jednoduché nastavení. Aplikace by měla na vstupu zadáno interval mezi jednotlivým sejmutím obrazovky. Pokud by během tohoto intervalu nebylo dokončeno vyhledávání ikon v předešlém sejmutém obrázku, následující obrázek by byl sejmut až v okamžiku dokončení vyhledávání. Jinak by vždy k dalšímu sejmutí obrazovky došlo v okamžiku zadaného intervalu.

**Poznámka*: Další nevýhodou těchto možností je aktualizace seznamu zakázaných ikon. Přidání nové sady zakázaných ikon představuje nelehkou operaci. Student samozřejmě nesmí mít k seznamu ikon přístup, aby nedocházelo k její jakékoliv modifikaci.

9. Závěr

Cílem této práce bylo především vytvoření dvou aplikací použitelných při online testech k vyšetření chování studentů.

První krokem pro vytváření aplikace pro hlídání obrazovky studenta bylo navrhnutí jednoduchých metod pro uložení sejmutého obrázku tak, aby v konečném důsledku nedocházelo k redundanci irelevantních méně podezřelých obrázků. Patřilo zde hledání změn v oblasti hlavního panelu, hledání možného okraje okna či konečné dohledání v oblasti změn. Za zmínku stojí také možnost ukládat obrázky sejmuté plochy podle spuštěných procesů, vytížení procesoru či využití paměti počítače. Druhým krokem bylo nalézt nejvhodnější formát pro uložení obrázků tak, aby z nich bylo možné později jednoznačně určit, zdali se v nich vyskytují zakázané ikony. V neposlední řadě jsme se zabývali možnostmi odesílání obrázků studenta. Bylo vyzkoušeno několik způsobů, z nichž každý má výhody či nevýhody pro různé situace.

Při vytváření aplikace sloužící pro vyhodnocení chování studenta byly vyzkoušeny jednoduché metody vyhledávání ikon, umožňující jednoznačné nalezení ikon v obrázcích, které se ukázaly být velmi neefektivní. Oproti nim se ukázala být podstatně vhodnější námi navrhnutá metoda s předzpracováním ikon i obrázků.

Výsledkem celé práce jsou dvě plně funkční aplikace. První pro ukládání podezřelých obrázků zachycených na obrazovce studenta vykonávajícího online test. A druhá aplikace, umožňující zkoušející osobě zobrazit podrobnosti o jednotlivých testech studentů a zvýraznit či zobrazit nepovolené zakázané ikony v jednotlivých podezřelých obrázcích. Vyhledávání ikon jednoznačně určuje, zdali se některá ze seznamu zakázaných ikon v obrázcích vyskytuje či nikoliv a probíhá se 100% úspěšností.

Posledním výsledkem práce, který stojí jen za zmínku, je sada zakázaných a ořezaných ikon použitelných při vyhledávání.

Literatura

[1] BĚHÁLEK, Marek. *Programovací jazyk C#.* [s.l.], [2007?]. 144 s. Oborová práce. Dostupný z WWW: <http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text.pdf>.

[2] *Wikipedie : Otevřená encyklopedie* [online]. 2009 [cit. 2009-04-06]. File Transfer Protocol Dostupný z WWW: <http://cs.wikipedia.org/wiki/File_Transfer_Protocol>.

[3] KAČMAŘÍK, Vojtěch. *Výuková podpora předmětu Internetové technologie*. [s.l.], 2006. 65 s. VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Vedoucí bakalářské práce Ing. Tomáš Tureček. Dostupný z WWW: <http://www.cs.vsb.cz/turecek/vyuka/via/data/diplomky/kacmarcik/prace.pdf>.

[4] ŽÁRA, Jiří. *Moderní počítačová grafika: Kompletní průvodce metodami 2D a 3D grafiky*. 2. přepracované a rozšířené vyd. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

[5] DRAYTON , Peter, ALBAHARI, Ben , NEWARD, Ted. *Csharp v kostce : Pohotová referenční příručka*. [s.l.] : [s.n.], 2003. 788 s. Moderní programování. ISBN 80-247-0443-9.

[6] *Microsoft Developer Network* [online]. c2009 [cit. 2009-02-04]. Dostupný z WWW: <http://msdn.microsoft.com>.

[7] *Csharp tutorial* [online]. c2003-2008 [cit. 2008-11-24]. Dostupný z WWW: <http://www.java2s.com/Tutorial/CSharp/CatalogCSharp.htm>.

Přílohy

A. Obsah přiloženého CD

Adresář	Obsah adresáře
/text/	Text této práce v elektronické podobě
/hlídaní studenta/zdroj/	Zdrojové kódy aplikace „Hlídání studenta“
/hlídaní studenta/dokumentace/	Programátorská dokumentace a uživatelská příručka aplikace „Hlídání studenta“
/hlídaní studenta/instalace/	Instalační soubory aplikace „Hlídání studenta“
/vyhodnocení/zdroj/	Zdrojové kódy aplikace „Vyhodnocení“
/vyhodnocení/dokumentace/	Programátorská dokumentace a uživatelská příručka aplikace „Vyhodnocení“
/vyhodnocení/instalace/	Instalační soubory aplikace „Vyhodnocení“
/ikony/	Sada zakázaných ikon připravených k vyhledávání
/test/	Testovaná data, ukázky výsledků testování

**Poznámka:* implementace aplikací byla provedena za pomocí zdrojů uvedených v seznamu použité literatury ([1], [5], [6], [7]).